Interactive, Computation Assisted Design Tools

Akash Garg

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2020

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Realistic modeling, rendering, and animation of physical and virtual shapes have matured significantly over the last few decades. Yet, the creation and subsequent modeling of three-dimensional shapes remains a tedious task which requires not only artistic and creative talent, but also significant technical skill. The perfection witnessed in computer-generated feature films requires extensive manual processing and touch-ups. Every researcher working in graphics and related fields has likely experienced the difficulty of creating even a moderate-quality 3D model, whether based on a mental concept, a hand sketch, or inspirations from one or more photographs or existing 3D designs. This situation, frequently referred to as the content creation bottleneck, is arguably the major obstacle to making computer graphics as ubiquitous as it could be. Classical modeling techniques have primarily dealt with local or low-level geometric entities (e.g., points or triangles) and criteria (e.g., smoothness or detail preservation), lacking the freedom necessary to produce novel and creative content.

A major unresolved challenge towards a new unhindered design paradigm is how to support the design process to create visually pleasing and yet functional objects by users who lack specialized skills and training. Most of the existing geometric modeling tools are intended either for use by experts (e.g., computer-aided design [CAD] systems) or for modeling objects whose visual aspects are the only consideration (e.g., computer graphics modeling systems). Furthermore, rapid prototyping, brought on by technological advances 3D printing has drastically altered production and consumption practices. These technologies empower individuals to design and produce original objects, customized according to their own needs. Thus, a new generation of design tools is needed to support both the creation of designs within the domain's constraints, that not only allows capturing the novice user's design intent but also meets the fabrication constraints such that the designs can be realized with minimal tweaking by experts.

To fill this void, the premise of this thesis relies on the following two tenets:

Tenet 1 users benefit from an interactive design environment that allows novice users to continuously explore a design space and *immediately* see the tradeoffs of their design choices.

Tenet 2 the machine's processing power is used to assist and guide the user to maintain constraints imposed by the problem domain (e.g., fabrication/material constraints) as well as help the user in exploring feasible solutions close to their design intent.

Chapter 2 provides an overview of some of the related considerations and approaches in regard to computational design. These approaches enhance the creative process by marrying creative exploration of a solution space assisted with numerical computation. The key insight is that interactivity

is a crucial element for any design process, as it serves to build intuition, increase productivity as well as increase quality [Doherty and Thadhani, 1982]. Interactive tinkering within the design space allows even novice users to build a mental model of the solution space. While humans are adept at navigating a large discontinuous space they fail to keep track of nuanced details imposed by the external constraints in the problem domain. Typically these constraints have numerical specifications (e.g., objects must have a certain weight/thickness to ensure fabrication) and thus can be easily solved by the processing power available on modern machines. Numerical specifications however, provide little room for creative exploration and thus they are best served as a guide by the computer to assist users when needed while leaving the user's mind unencumbered to focus on the overall design intent.

Finding the appropriate balance between interactive design tools and the computation needed for productive workflows is the problem addressed by this thesis. This thesis makes the following contributions:

1. We take a close look at thin shells–materials that have a thickness significantly smaller than other dimensions. Towards the goal of achieving interactive and controllable simulations we realize a particular geometric insight to develop an efficient bending model for the simulation of thin shells. Under isometric deformations (deformations that undergo little to no stretching), we can reduce the nonlinear bending energy into a cubic polynomial that has a linear Hessian. This linear Hessian can be further approximated with a constant one, providing significant speedups during simulation. We also build upon this simple bending model and show how orthotropic materials can be modeled and simulated efficiently.

2. We study the theory of Chebyshev nets–a geometric model of woven materials using a two-dimensional net composed of inextensible yarns. The theory of Chebyshev nets sheds some light on their limitations in globally covering a target surface. As it turns out, Chebyshev nets are a good geometric model for wire meshes, free-form surfaces composed of woven wires arranged in a regular grid. In the context of designing sculptures with wire mesh, we rely on the mathematical theory laid out by Hazzidakis [Hazzidakis, 1879] to determine an artistically driven workflow for approximately covering a target surface with a wire mesh, while globally maintaining material and fabrication constraints. This alleviates the user from worrying about feasibility and allows focus on design.

3. Finally, we present a practical design tool for the design and exploration of reconfigurables, defined as an object or collection of objects whose transformation between various states defines its functionality or aesthetic appeal (e.g., a mechanical assembly composed of interlocking pieces, a transforming folding bicycle, or a space-saving arrangement of apartment furniture). A novel space-time collision detection and response technique is presented that can be used to create an interactive workflow for managing and designing objects with various states. This work also considers a graph-based timeline during the design process instead of the traditional linear timeline and shows its many benefits as well as challenges for the design of reconfigurables.

1.1 Organization

This thesis is organized around three core chapters: Chapter 3 exploits a geometric insight in order to create a computationally inexpensive and expressive model for the simulation of thin shells [Garg

et al., 2007]. Building on the insights for orthotropic materials presented in [Garg *et al.*, 2007], Chapter 4 couples a computational framework into an interactive design workflow to assist the design of sculptures made of wire mesh [Garg *et al.*, 2014]. Finally, Chapter 5 addresses the design of reconfigurable objects that exist and transition between various different states [Garg *et al.*, 2016]. The last chapter organizes some concluding thoughts on the work presented in this thesis as well as proposes several directions for future investigation.

Chapter 2

Related Work

Fundamentally, design is about choice, and the manner in which meaningfully distinct choices are presented to users is a central determining factor in the success of any design process [Buxton, 2007]. The development and evaluation of alternative choices create a design loop that is continually reformulated. The feedback loop is widely held to be an integral part of good design. Each iteration of this design loop makes additional considerations, imposes additional constraints, and requires further experimentation in order to refine the designer's understanding of the space of possible designs [Terry and Mynatt, 2004; Sternberg, 1999; Löwgren, 2006]. The following section strives to highlight some relevant work that address several design considerations and approaches the problem of dealing with constraints in an elegant manner.

Structural Considerations To fill the gap between digital geometry and understanding design choices, We have to fundamentally consider the *structure* of shapes which considers a higher-level characterization of 3D shapes. Even seemingly simple objects contain a multitude of complex relations. Such relations (or constraints) arise from various practical considerations, which can be categorized as: semantic considerations (e.g., table-tops are horizontal); functional considerations (e.g., chair legs support and keep the chair stable); and fabrication or economic considerations (e.g., repeated object parts are easier to manufacture). Violating such structural constraints during model creation leads to implausible or unnatural results. Furthermore, as digital prototyping and physical fabrication becomes commonplace synthesized objects are held to a higher standard; they must be functional as well as physically realizable.

To reduce the burden on designers and artists, we need tools that can assist designers in recognizing structural properties of a shape, understanding shape interdependencies, and fabricate those shapes efficiently. In order to create such tools, we must understand the parts of a shape and the relationships between those parts. Parts can be specified by the user as a set of connected triangle meshes [Jain *et al.*, 2012; Zheng *et al.*, 2013] or from a scene graph hierarchy [Fisher *et al.*, 2011]. Shape segmentation approaches can also be applied that are useful for identifying parts of a fixed model [Gal *et al.*, 2009a; Mitra *et al.*, 2010; Bokeloh *et al.*, 2010; Kalojanov *et al.*, 2012; Shamir, 2008]. Identifying relations among parts can be treated as the problem of finding global correspondences between parts. Significant geometric and topological variances in shapes make this problem rather challenging. In recent work on CAD sketching interfaces, the user is allowed to specify geometric constraints that are verified and maintained the system [Zeleznik *et al.*, 2006; Igarashi *et al.*, 1999]. Constraints can also be fixed independent of the geometry, such as connectivity,

center of mass (balance) [Prévost *et al.*, 2013], and constraints of maximum stress within the material [Umetani *et al.*, 2012a; Whiting *et al.*, 2012]. Constraints within parts can also be learned from the data [Mitra and Pauly, 2008; Bokeloh *et al.*, 2010].

Symmetry & Structure Symmetry is a purely geometric notion and yet it plays a crucial role in identifying semantic information and structural analysis of man-made objects. Several methods have been proposed for analyzing symmetry in objects [Mitra *et al.*, 2012]. Symmetry is an attractive model for structure and shape analysis since it can be used to create abstractions from the concrete shape and be used to formulate complex, high-level structural assumptions about larger classes of geometry. Symmetry for example has been used to complete partial shapes [Thrun and Wegbreit, 2005] as well as create perfectly symmetric geometry [Mitra *et al.*, 2007].

Symmetry can also be used as an invariant for controlling object deformation. The iWires system [Gal *et al.*, 2009a] is based on the assumption that a free-form edit of an object should maintain the original symmetry properties of the input shape. The interaction and structure detection is based on line features called 'wires'. Similar abstractions have been used by Bokeloh et al. [2011; 2012], in order to deform objects in a (symmetry) structure preserving manner. Finally, symmetry in many man-made objects is not only expected but is intentionally designed to perform the same function. Hence, detected and modeled symmetries can provide first cues into functional analysis.

Functional & Fabrication Considerations The inherent geometry of an object can tell us a lot about its structure and how it can/should be edited, but says very little about its function. Often times, designers wish to preserve the functional aspects of objects during the design process. This becomes paramount as we move towards a design paradigm supported by on-demand fabrication technology enabled by 3D printing. Functional analysis is not only useful for the immediate understanding of shapes and their interactions but plays a prominent role when it comes to fabrication. Coros et. al. [2013] extend Mitra's work [Mitra *et al.*, 2010] on understanding mechanical assemblies to design characters actuated with a set of mechanical assemblies. Their work allows the user to input a depiction of a motion curve for each part of the mechanical toy. The system then samples the parameter space of all available mechanical gears in a given dataset and optimizes the arrangement of the gears to match the input motion.

3D printing poses a new set of problems in modeling such as the presence of support structures or even in the case of Prévost et. al. [2013] if shapes are guaranteed to stand on their own weight. Prévost et. al. present a system where the user drives the major deformations while the system optimizes the objects once 3D printed to stand under their own weight. An interactive user interface is presented that optimizes the interior to balance the center of mass while the user edits the overall surface of the shape using standard modeling tools.

Deformable objects however can complicate the fabrication process and require more sophisticated tools. Skouras et. al. [2012a] showcase a design loop for computationally intensive problems that have no hope for an interactive design loop. Their work focuses on solving the inverse problem of finding a suitable rest configuration for a latex balloon such that when inflated it takes the form of the original design intent. Such methods allow the designer to model highly unintuitive shapes that undergo nonlinear deformations that eventually meet the designer's original goal shape.

Complex design spaces do not always have to be dictated by physical laws; most often a design space is complex to navigate due to the semantic nature of the domain. Yu et. al. [2011] considered the inverse problem by synthesizing furniture layouts based on a set of given examples. Their

approach extracts key relationships between objects from a given set of examples. Then starting from a randomized arrangement, iteratively samples the domain until a desired synthesized scene is found. The search space for all combinations of furniture layouts in a scene is prohibitively large and also a unique global optimum is unlikely to exist or even be desired. Keeping these properties in mind, Yu et. al. formulate the optimization as a monte-carlo method which involves stochastically sampling the search space while reducing the cost-terms that measure deviation from the relationships extracted from the input examples. At each iteration, the placement and orientation of each object in the room is changed based on a normal Gaussian distribution. The cost of the new configuration is computed and the metropolis algorithm is used to determine if the new state is acceptable or not. The iteration continues until a desired outcome is reached.

Talton et. al. [2011] and Vanegas et. al. [2012] use the markov-chain-monte-carlo (MCMC) method for optimally sampling high-dimensional spaces when considering procedural design. Procedural tools can be quite powerful especially when creating organic shapes like trees. However, controlling the desired outcome from procedural grammars is quite challenging. There is a non-intuitive relationship between the input parameters and the generated output. Talton et. al. describe a similarity metric that compares the generative geometry constructed from an L-grammar for the creation of skylines and trees–he uses a cost function that compares against the silhouette of a input sketch. Vanegas et. al. use the MCMC approach by letting the user specify output constraints such as road thickness, building heights, distance to parks, etc. in order to procedurally generate cities. Both approaches rely on simulated annealing approaches that sample the domain space in order to find good global optimum solutions. The inverse design paradigm allows allows users to work at a high level of abstraction and evaluate the generated design interactively, freeing them to focus on the design task of constraint considerations instead of focusing on the actual procedural or physical parameters needed to make their designs a reality.

Interactive Design Although inverse design helps to abstract the design process, it does not help when the design process is open-ended or exploratory. Here the user begins the design process with an under-specified goal and the precise final model is established through experimentation. Design processes are highly nonlinear: candidate designs are considered and rejected as the design space is explored. Since the final product is not comprehended in full detail at the onset, progress is made opportunistically and serendipitously. Many actions taken by the designer serve to refine the user's understanding of the space of possible designs than moving directly toward a final product.

To facilitate the design process, interactive design techniques are better suited to creating complex designs by building intuition. Intuition comes from interactivity and immediate feedback [Shneiderman, 2007]. This becomes especially useful when exploring large complex design spaces. The difference when compared to routine design is that one wants to actually explore what is possible. Without interactive feedback from the system, it is tough to build an intuition about the space of designs [Hauschild and Karzel, 2011; Dodgson *et al.*, 2005]. It is well established that development and evaluation of alternatives contribute to the continual elaboration and reformulation of the underlying design task. The feedback loop is widely held to be an integral part of good design [Llach, 2015].

This interactive workflow is particularly well suited for large design spaces that have many nonlinear constraints; for instance, designing with planar-quad meshes efficiently and interactively has been a challenging problem where interactivity has been fruitful in developing novel and compelling shapes [Yang *et al.*, 2011; Zhao *et al.*, 2012; Deng *et al.*, 2013]. The designer is able to locally explore a design by using handles. The system automatically gives immediate feedback

by providing an approximate surface that closely matches the designer's edit–by moving along the tangent space of the constraint manifold. The work of Yang et. al. [2011] use eigenvectors of the constraint Hessian, which provides linear basis along which edits can be made. Once the editing is done, the design is projected back onto the constrained manifold using a minimization routine, e.g., SQP (sequential quadratic programming). Theoretically, the minimization procedure is fast since the modified mesh should be close to the projection. This setup has subsequently been extended to handle curve-based model deformations [Zhao *et al.*, 2012], and also deformations restricted to local modifications based on sparsity considerations [Deng *et al.*, 2013].

Interactive workflows that rely on fast computation are often challenging to achieve in a design space that is not only governed by geometric constraints but also physical constraints. Typically a dynamical system must integrate the system's degrees of freedom through a series of small incremental timesteps. This procedure is typically computationally expensive for complex nonlinear physical systems, especially those with interacting elements. Modeling physical materials and their properties can be a non-trivial task, especially when considering their microstructure. One popular method in the applied mathematics community to handle complex heterogeneous materials is, *homogenization*. Homogenization attempts to find an equivalent material model that does not require the need to model each individual microstructure. This model characterizes the average mechanical behavior as well as represent the effect of composite material heterogeneities [Oleinik, 1984; Mei and Vernescu, 2010].

Homogenization has also been particularly useful in the study of cloth simulation [Saito, 2013; Hearle *et al.*, 2001] because the behavior of textiles is greatly affected by their microstructure [Nadler *et al.*, 2006]. Capturing this microstructure behavior is important to capture the subtle nuances of the simulating fabric. Nadler et al. [2006] present a multi-scale model that treats the macroscale behavior as a continuum while modeling the microstructure as a pair of overlapping yarns as extensible/inextensible elastica. Nadler's work is further extended by Parsons et. al [2010] and Xia et. al. [2011] who also combine continuum models with the microstructure properties of fabric for accurate numerical simulation of certain classes of fabrics, such as Kevlar. Instead of relying on multiscale approaches, Cirio et. al. [2014] discretize each yarn in the fabric in order to capture both the micro and macro scale behavior. Their results, although encouraging, are too computationally expensive for use in the realm of interactive design.

The challenge remains how to capture the subtle nonlinear behavior exhibited by a material's microstructure at interactive rates that can be used in design applications. Instead of attempting to run simulations at interactive rates, many data-driven approaches run expensive simulations offline and then use that data to simulate materials at interactive rates when needed. One common approach in data-driven methods is to run low-resolution simulations at interactive rates and add the high-resolution details based on previously run high-resolution simulations [Wang *et al.*, 2010; Kavan *et al.*, 2011; Feng *et al.*, 2010]. While these approaches dramatically increase the realism of the simulations, the resulting motion is still reminiscent of a low-resolution simulation, i.e., the microstructure of the material does not play a role in affecting the macro scale behavior of the material. In contrast, James and Fatahalian [2003] tabulate the arbitrary nonlinear dynamics of a deformable system and are able to achieve real-time cloth simulation by navigating a database of cloth trajectories residing in a precomputed subspace. In the same vein, Twigg and James [2007; 2008] exploit dynamic bifurcations (arising from collisions) for control. Their work explores contact ambiguity to create animations which converge to a desired final state. However, data-driven methods are only as good as the density of phase-space samples they are able to precompute. Kim et.

al. [2013] show that very large scale sampling of the state-space is possible and able to produce complex dynamic behavior for interactive playback.

Although data-driven approaches show promise, a concern with these approaches are the limits available in real-time memory footprint, especially for more complex dynamical systems, e.g., fluids. Barbic and James [2005] and An et. al. [2008] use model-reduction to fully capture complex systems. These subspace integration methods form a reduced basis, for example using modal analysis [Thomson, 1996]. Linear modal analysis [Shabana, 1990] is one way to form a reduced basis, which provides best deformation basis for small deformations away from the rest pose. Intuitively, modal basis vectors are directions into which the model can be pushed with the smallest possible increase in elastic strain energy. Barbic and James extend past linear modal analysis to the nonlinear regime by forming a subspace using modal derivatives. They are able to show that good performance as well as accuracy can be increased by using modal derivatives instead of linear modal analysis. Subspace methods greatly reduce the cost of nonlinear simulation, however the lower-dimensional basis that is used usually has global support and as such is unable to capture localized deformations. Harmon and Zorin [2013] augment the subspace basis vectors at runtime such that they can respond to localized deformations that occur as a result of external stimuli (e.g., collisions).

Forming a lower-dimensional basis for deformation is a hard open problem in solid mechanics, and there exist no algorithms for automatic proven-quality global deformation basis generation under general forcing [Barbič and James, 2005]. Subspace methods simplify the degrees of freedom in the system in order to capture complex behavior. Another useful approach to achieving real-time physical simulations is by linearizing the physical model itself. Linearization methods have been quite useful to gain intuitive interactive understanding of the design space not only for geometric constraints but also when designing objects with physical constraints. Interactive design workflows for the design of garments and furniture [Umetani *et al.*, 2011; Umetani *et al.*, 2012a] both require the linearization of the physics of the underlying model instead of the geometric constraints. The physics here acts as a means to understand relationships between components that make up the entire object.

The work of Umetani and colleagues [2011] relies on using sensitivity analysis, which attempts to provide (first-order) insight into the relationships between design and performance in either one-off calculations or offline optimizations. Linear sensitivity is often insufficient for highly nonlinear physics and thus edits are coupled with simulation with progressive nonlinear modeling. Moving Least Squares (MLS) is used to approximate the result while of the user's edits at interactive rates. When the user pauses, the system solves the full nonlinear system of equations in order to accurately capture the effects of the user's edits. Umetani et. al. [2012a] then extend their use of linearized sensitivity analysis for interactive furniture design. This time the linearized sensitivity matrices are used to provide suggestions. Linearization of forces are used to pick good search directions which can help the user "autocomplete" a design task or receive suggestions for designs they have not considered.

Linearization of the design space however does not always result in a good approximation, especially when the energy function is non-smooth. Merrell et. al. [2011a] explored interactive suggestion based design techniques for furniture arrangement. Unlike the approach of Yu et. al. [2011], the relationships between parts are encoded as energy functions. While characterizing the valid space itself is difficult, exploring high degree of freedom design spaces is challenging as the valid regions may be disjoint, forming islands, or have narrow connection pathways among valid

spaces, posing further challenges. In such cases, linearized techniques can fail to find plausible configurations, even when they exist. To this end, Merrell et. al. [2011a] consider an interactive suggestion oriented design tool based on the MCMC (markov-chain-monte-carlo) method. Their work first allows the user to manipulate the scene, and then the system samples various potential solutions offering them as suggestions which the user can select or ignore. The key advantage to MCMC methods is that computation can be halted when resources have expired. There is no need to find a global optimum. Merrell implemented the method of parallel tempering on the GPU and is able to create suggestions in about one second. The massive parallel ability and halting of MCMC makes it a suitable approach to suggestion-based design.

Some large complex design spaces however cannot be sampled efficiently or do not have effective parameterizations. Talton et. al. [2009] show this in the case of organic objects like trees and the shapes of human bodies. Here organizing these datasets with several clustering methods (e.g., PCA) does not do an effective job, whereas humans are adept at deciphering semantic information about shapes very quickly. A semantic approach also enables us to parameterize methods and make sense of them without having knowledge a-priori about the structure of the objects. They are in a sense inferred by users of the system.

The role of having human intuition guide the design process in various and often subtle ways plays a central theme in the remainder of this thesis.

Chapter 3

Cubic Shells

3.1 Introduction

Many animation applications require simple and efficient simulation of a general class of elastic surfaces. This class includes objects that are (a) flat (plates) or curved (shells) in their undeformed state, (b) flexible or nearly rigid and (c) isotropic or anisotropic in their response to bending. In mesh-based simulation, *hinge-based* methods are preferred for their simplicity and economy of computation. Considering every two triangles meeting at an edge to be a bending hinge, such methods require that some function of the dihedral angle is subject to a restoring force. Empirically, hinge-based models are known to work well for isotropic materials, and for geometric models of anisotropy based on Euler's curvature formula [Baraff and Witkin, 1998; Volino and Magnenat-Thalmann, 2006b]. For a survey of bending models used in animation, see [Thomaszewski and Wacker, 2006].

Contributions. We present what we believe is the simplest and most efficient hinge-based bending model to encompass the spectrums (a), (b), and (c), at the cost of restricting our attention to *quasi-inextensible* surfaces, *i.e.*, surfaces that prefer to bend much more than to stretch. Our model is based on a mathematical analysis of the hinge-based approach resulting in two main insights:

First, we introduce a hinge-based bending model that preserves a key property of the smooth setting: the bending energy of a thin shell is *cubic* under isometric deformations. To understand the consequences of this statement, consider that bending energy is in general a highly nonlinear functional of the surface position; therefore, the implementation of implicit solvers for thin shells involves relatively complex derivation and costly computation of the nonlinear bending force Jacobian. However, under the assumption of quasi-isometry, we show that implementation can be reduced to a one-time precomputation of an approximate Jacobian matrix, and implicit time stepping routines can use an inexact Newton method for significant performance gains. Shells simulated with this method conserve both linear and angular momenta.

Second, we treat anisotropy, where mechanical response depends on the direction of applied strain. We present a bending model for *orthotropic* materials, exposing an important physical parameter—the *shear modulus*—which affects the *drapability* of a fabric [Sidabraite and Masteikaite, 2002]. Perhaps due to their small stencil, hinge-based models in computer animation have thus far eluded the incorporation of orthotropic response.

Our work builds on a foundation laid out by Bergou *et al.* [Bergou *et al.*, 2006], who demonstrated

that in the special case of isotropic thin *plates* undergoing isometric deformations, the bending energy is quadratic, and correspondingly the forces are linear, in mesh positions; see also Volino and Magnenat-Thalmann [Volino and Magnenat-Thalmann, 2006b] for a linear bending force. In contrast, we consider inextensible thin *shells*, whose bending energy is cubic, with correspondingly quadratic forces.

Overview. By first considering *isotropic* bending, we expose the cubic nature of bending energies in the smooth (§3.2.1) as well as discrete settings (§3.2.2), establishing the *cubic hinge* (§3.3) as our discrete building block. Next, we introduce *orthotropy* (§3.4) and show that it is captured by a scalar hinge stiffness (§3.4.1). Finally, we describe the efficient implementation of orthotropic cubic shells (§3.5–§3.6), and discuss a battery of experiments demonstrating the efficiency and efficacy of our model (§3.7–§3.8).

3.2 Isotropic bending

3.2.1 Smooth setting

Consider a surface deformed away from its natural (*undeformed*) shape. To this deformation we associate the isotropic bending energy

$$E_{\mathrm{b}}(\mathbf{x}) = \frac{1}{2}\int_S (H - \overline{H})^2 \mathrm{d}A\,. \tag{3.1}$$

This is the integral, over the undeformed surface, of the squared change in mean curvature. Here $\mathbf{x}$ is the position of the deformed surface, and H is the mean curvature function of the deformed surface. A bar (*e.g.*, $\overline{H}$) denotes the corresponding quantity evaluated on the undeformed surface. Whereas we consider arbitrary undeformed shapes, the special case of flat undeformed shapes ($\overline{H} = 0$) was explored by Bergou *et al.* [Bergou *et al.*, 2006].

Although not immediately apparent from (3.1), $E_{\mathrm{b}}(\mathbf{x})$ is actually a *cubic polynomial* in $\mathbf{x}$ under *isometric deformations*, *i.e.*, if the surface is allowed to bend but not to stretch. To see this, rewrite (3.1) as

$$E_{\mathrm{b}}(\mathbf{x}) = \frac{1}{2}\int_S \left(\langle \mathbf{H}, \mathbf{H}\rangle - 2\langle \mathbf{H}, \overline{H}\hat{\mathbf{n}}\rangle + \overline{H}^2\right) \mathrm{d}A\,. \tag{3.2}$$

Here $\langle\cdot,\cdot\rangle$ denotes the standard inner product in 3-space, and $\mathbf{H} = H\hat{\mathbf{n}}$ stands for the mean curvature *normal*, a vector parallel to the surface's unit normal, $\hat{\mathbf{n}}$. Bergou *et al.* observed that the mean curvature vector can be expressed as $\mathbf{H} = \Delta\mathbf{x}$, *i.e.*, the surface's intrinsic Laplacian applied to the position of the surface. For isometric deformations of the surface, two facts follow: (F1) Since Δ is intrinsic, $\mathbf{H}$ is *linear in surface position*, thus $\langle \mathbf{H}, \mathbf{H}\rangle$ is quadratic in $\mathbf{x}$ [Bergou *et al.*, 2006]. (F2) The surface normal is quadratic[1] in $\mathbf{x}$; therefore, $\langle \mathbf{H}, \overline{H}\hat{\mathbf{n}}\rangle$ is cubic. It follows that $E_{\mathrm{b}}(\mathbf{x})$ is a cubic polynomial in $\mathbf{x}$, because $\langle \mathbf{H}, \overline{H}\hat{\mathbf{n}}\rangle$ is cubic, $\langle \mathbf{H}, \mathbf{H}\rangle$ is quadratic, and $\overline{H}^2$ does not depend on $\mathbf{x}$. While Bergou *et al.* discussed (F1) for thin plates ($\overline{H} = 0$), (F2) is unique to shells ($\overline{H} \neq 0$) and presents our central technical challenge.

The *cubic shells* idea is to develop a discrete analogue of this picture.

[1] To see that $\hat{\mathbf{n}}$ is quadratic, write $\hat{\mathbf{n}} = \frac{\partial \mathbf{x}}{\partial u} \times \frac{\partial \mathbf{x}}{\partial v}$, and observe that orthonormality of tangent vectors, $\frac{\partial \mathbf{x}}{\partial u}$ and $\frac{\partial \mathbf{x}}{\partial v}$, is preserved under isometric deformations.

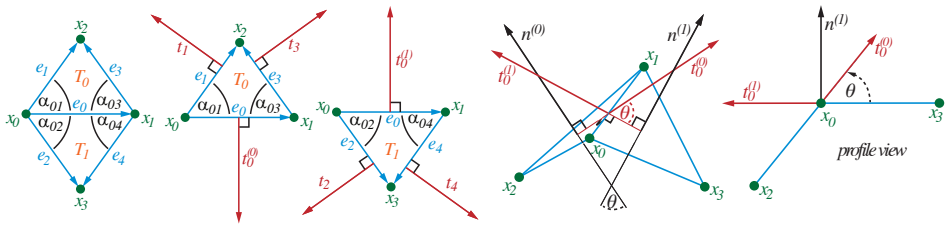


Figure 3.1: *Top-left:* Hinge stencil for an interior edge. *Top-middle and -right:* Perpendicular vectors used in the computation. *Bottom:* Hinge in 3-space with corresponding labels.

3.2.2 Discrete setting

Consider a triangle mesh whose shape before deformation is given by the vector of vertex positions $\overline{\mathbf{x}}$. When the mesh is deformed to position $\mathbf{x}$, the usual hinge-based bending energy[Baraff and Witkin, 1998; Bridson *et al.*, 2003; Grinspun *et al.*, 2003; Bergou *et al.*, 2006; Thomaszewski and Wacker, 2006] is given by[2]

$$E_{\mathrm{b}}(\mathbf{x}) = \frac{1}{2}\sum_i \frac{3|\overline{\mathbf{e}}_i|^2}{\overline{A}_i}\left(2\sin\frac{\theta_i - \overline{\theta}_i}{2}\right)^2 . \tag{3.3}$$

Here the sum is taken over all interior edges, $\overline{A}_i$ denotes the combined area of the two triangles incident to edge $\overline{\mathbf{e}}_i$, and θ_i denotes the dihedral angle at edge i.

Our goal is to establish an important, and previously overlooked, property of (3.3): under isometric deformations, it is a cubic polynomial in $\mathbf{x}$. This observation will expose a much more efficient implementation than is directly evident from (3.3); for details on this implementation, refer directly to §3.5.

The discrete energy (3.3) can be written as a sum over contributions from every hinge (indexed by i). Using the identity $2\sin^2 u = 1 - \cos 2u$, with $u = (\theta - \overline{\theta})/2$, we arrive at an expression of energy associated to the i^{th} hinge:

$$E_{\mathrm{b}}(\mathbf{x})_i = \frac{3|\overline{\mathbf{e}}_i|^2}{\overline{A}_i}(1 - \cos(\theta_i - \overline{\theta}_i)) . \tag{3.4}$$

In the following section, we prove that for isometric deformations this hinge energy is cubic in $\mathbf{x}$.

3.3 The cubic hinge

Our derivation uses the geometric construction in Figure 3.1, which defines *local indices* of the triangles, $\{T_0, T_1\}$, edge vectors, $\{\mathbf{e}_0, \ldots, \mathbf{e}_4\}$, and vertex positions, $\{\mathbf{x}_0, \ldots, \mathbf{x}_3\}$, associated to a hinge. Note the construction of *perpendiculars* $\mathbf{t}_i$: each edge vector $\mathbf{e}_i$ is rotated 90 degrees, on the plane of the triangle, to point outwards (thus $|\mathbf{t}_i| = |\mathbf{e}_i|$ and $\langle \mathbf{t}_i, \mathbf{e}_i \rangle = 0$). Since $\mathbf{e}_0$ is associated to two triangles, we construct two perpendiculars, $\mathbf{t}_0^{(0)}$ and $\mathbf{t}_0^{(1)}$, on the planes of triangles T_0 and T_1 respectively.

[2]For planar rest state ($\overline{\theta} = 0$) we have $\lim_{\theta\to 0} 2\sin\theta/2 = \lim_{\theta\to 0} 2\tan\theta/2 = \theta$, so that models that use $2\sin\theta/2$, $2\tan\theta/2$, or θ coincide in the limit of an appropriate refinement sequence.

Expressing the perpendiculars $\mathbf{t}_0^{(0)}$ (resp. $\mathbf{t}_0^{(1)}$) in the edge basis $\{\mathbf{e}_1, \mathbf{e}_3\}$ (resp. $\{\mathbf{e}_2, \mathbf{e}_4\}$) we obtain

$$\mathbf{t}_0^{(0)} = -\cot\alpha_{03}\mathbf{e}_1 - \cot\alpha_{01}\mathbf{e}_3 \,, \tag{3.5}$$

$$\mathbf{t}_0^{(1)} = -\cot\alpha_{04}\mathbf{e}_2 - \cot\alpha_{02}\mathbf{e}_4 \,. \tag{3.6}$$

Under isometric deformation, interior angles remain constant, therefore $\mathbf{t}_0^{(0)}$ and $\mathbf{t}_0^{(1)}$ are *linear expressions* in the position of the mesh. Furthermore, the geometry of Fig. 3.1-bottom reveals that[3]

$$\cos\theta = \frac{-\langle \mathbf{t}_0^{(0)}, \mathbf{t}_0^{(1)} \rangle}{|\mathbf{e}_0|^2}, \quad \text{and} \quad \sin\theta = -\frac{\beta[\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2]}{|\mathbf{e}_0|^2} \,,$$

where $[\mathbf{e}_i, \mathbf{e}_j, \mathbf{e}_k]$ is the scalar triple product $(\mathbf{e}_i \times \mathbf{e}_j)\cdot\mathbf{e}_k$, and $\beta = \frac{1}{|\mathbf{e}_0|}(\cot\alpha_{01} + \cot\alpha_{03})(\cot\alpha_{02} + \cot\alpha_{04})$.

Expanding (3.4) by the identity $\cos(\theta - \overline{\theta}) = \cos\theta\cos\overline{\theta} + \sin\theta\sin\overline{\theta}$, substituting the above expressions for $\cos\theta$ and $\sin\theta$, and simplifying the resulting expression by using the isometry assumption ($|\overline{\mathbf{e}}_i| = |\mathbf{e}_i|$) we find that (3.4) equals

$$\underbrace{\frac{3}{\overline{A}_0}\left(|\overline{\mathbf{e}}_0|^2 + \langle \mathbf{t}_0^{(0)}, \mathbf{t}_0^{(1)} \rangle \cos\overline{\theta}\right)}_{\text{thin plate component}} + \underbrace{\frac{3\overline{\beta}}{\overline{A}_0}[\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2]\sin\overline{\theta}}_{\text{cubic term}} \,. \tag{3.7}$$

Under isometry, the energy associated to an individual hinge is a cubic polynomial in $\mathbf{x}$. Just as in the smooth picture, the discrete $E_\text{b}(\mathbf{x})$ is cubic. Note that for the special case $\overline{\theta} = 0$, we recover the quadratic bending thin *plate* energy of [Bergou *et al.*, 2006] simply by substituting (3.5) and (3.6) into (3.7).

3.4 Orthotropy

We now generalize our bending model to *orthotropic* materials—an important class of anisotropic materials whose elastic properties depend on the direction along which they are measured [Ventsel and Krauthammer, 2001]. A fully general linear elasticity model for deformable surfaces has six parameters (not counting the choice of anisotropy axes) some of which are hard to interpret intuitively. We focus on a more restricted *orthotropic* elasticity model whose four parameters have more intuitive meaning and appear to be useful for material behavior control in animation. Most common man-made materials are orthotropic, for example, cloth, plastic reinforced by fibers, sheet metal, and paper. Non-orthotropic thin materials are less common (*e.g.*, thin sheets obtained by cutting a 3D orthotropic material at an angle, or composite materials). For cloth, orthotropic approximation naturally matches most of the parameters of the Kawabata cloth evaluation system [Kawabata, 1980], a commonly used system for characterizing cloth properties, as explained below.

[3] The expression for $\sin\theta$ was derived from

$$\sin\theta = \frac{\langle \mathbf{t}_0^{(0)}, \mathbf{n}^{(1)} \rangle}{|\mathbf{e}_0|^2} = -\frac{[\mathbf{t}_0^{(0)}, \mathbf{t}_0^{(1)}, \mathbf{e}_0]}{|\mathbf{e}_0|^3} = -\frac{\beta[\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2]}{|\mathbf{e}_0|^2} \,,$$

where $\mathbf{n}^{(1)} = (|\mathbf{e}_0|/|\mathbf{e}_4 \times \mathbf{e}_2|)\mathbf{e}_4 \times \mathbf{e}_2$ is a scaled triangle normal. The last step is obtained by using (3.5) and (3.6) to replace $\mathbf{t}_0^{(0)}$ and $\mathbf{t}_0^{(1)}$, followed by observing that $\mathbf{e}_4 = \mathbf{e}_2 - \mathbf{e}_0$ and $\mathbf{e}_3 = \mathbf{e}_1 - \mathbf{e}_0$.

We are primarily interested in how these material parameters affect *bending*, rather than in-plane deformations. From the four parameters of orthotropic materials, one parameter can be eliminated if we assume that bending the surface along material directions does not have any effect on bending on the other direction (this corresponds to having zero Poisson ratio in the isotropic case). For simplicity we will assume that this is the case, and briefly discuss the role of the Poisson ratio at the end of this section.

The most obvious of the remaining three parameters are two Young's moduli, Y^0 and Y^1, which determine bending stiffness along two material directions. The third parameter is the *shear modulus*, G^{01}, which, for in-plane deformations, determines the resistance to shear. In the case of bending, the shear modulus allows for additional directional variability of bending stiffness. Specifically, bending stiffness at an angle α with respect to the material axis 0, is given by (up to a scale factor):

$$Y^0 \cos^4 \alpha + Y^1 \sin^4 \alpha + 2G^{01} \sin^2 \alpha \cos^2 \alpha \, . \tag{3.8}$$

Consider the plot of *directional stiffness* in Figure 3.2 as a function of α, for three materials all sharing $Y^0 = Y^1$, but with shear moduli $0.1K$, K, and $2K$. Note that for low shear, typical for cloth, the maximal to minimal bending stiffness ratio is 2. As shown in Figure 3.5, this has a significant effect on drapability, and *cannot* be achieved by a simple model using just two parameters. Similarly, the shear modulus affects resistance of cloth to *twisting*, even if $Y^0 = Y^1$, as twisting may lead to diagonal bending.

Directional stiffness is particularly important in the case of *quasi-inextensible* flat sheets. Such sheets have small Gaussian curvature (the product of the two principal curvatures), which implies that strong bending can be present only in a single direction.

In contrast to directional stiffness, forces arising from the interaction of *multiple* bending directions (*e.g.*, due to the Poisson ratio in the isotropic case) appear to be qualitatively less important. In fact, one can show that for directional stiffness this interaction results in effective increase in the

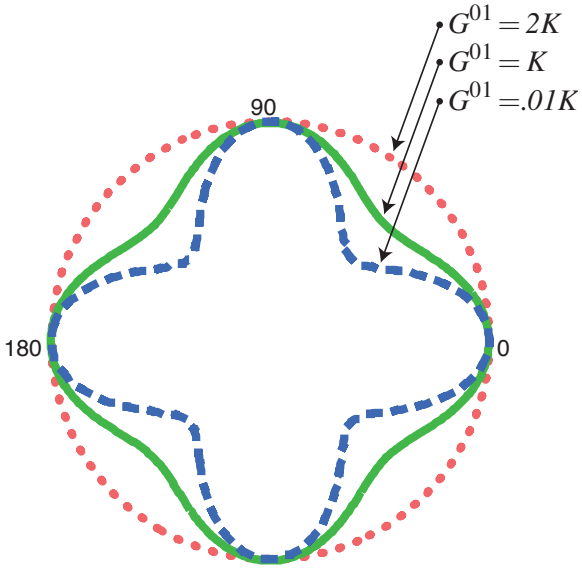


Figure 3.2: *Directional stiffnesses* as a function for several values of G^{01}: red (dotted) $G^{01} = 2K$, green (solid) $G^{01} = K$, and blue (dashed) $G^{01} = 0.1K$.

shear modulus. One can prove that in the simple case of bending of a plate with fixed boundaries the bending forces (unlike stretching) are independent of the Poisson ratio.

The Kawabata system characterizes cloth by its resistance to bending along warp and along weft directions, and by tensile, shearing and compressive stiffnesses measured as functions of deformation. In the orthotropic elasticity model, stiffnesses are assumed to be independent of the deformation, which appears to be a good approximation for qualitative modeling. Furthermore, plasticity effects are ignored, and the compressive and tensile stiffnesses are assumed to be equal. As pointed out in [Breen and Donald, 1994], linear elasticity provides a good approximation for Kawabata measurements for small deformations, although it ignores some of the subtler initial-resistance effects.

Given Kawabata measurements of a fabric, Y^1 and Y^0 can be inferred from directional bending stiffnesses and tensile measurements, and G^{01} from shear measurements. Alternatively, Y^1, Y^0, G^{01}, and the material axes may be directly controlled by an artist, adjusting parameters based on the notion that Y^1 and Y^0 determine resistance to bending along the two orthogonal material directions and G^{01} determines the resistance of the material to draping over an object.

Given the material parameters, and applying a standard formulation of Hooke's law for orthotropic materials, we can express the bending energy density of a deformed surface as a function of bending strain ϵ, using one additional material parameter $Y^{01} = \nu_{01}Y^1 = \nu_{10}Y^0$, where ν_{10} and ν_{01} are Poisson ratios (which always satisfy $\nu^{01}/Y^0 = \nu^{10}/Y^1$). The energy density is

$$c\left(Y^0\epsilon_{00}^2 + Y^1\epsilon_{11}^2 + 2Y^{01}\epsilon_{00}\epsilon_{11}\right) + 2hG^{01}\epsilon_{01}^2\,, \tag{3.9}$$

where $c = hY^0Y^1/(Y^0Y^1 - Y^{01}Y^{01})$, $h = \tau^3/12$, and τ is the sheet thickness. The bending strain is the shape operator [do Carmo, 1992] for surface, which for a quadratic bending energy reduces to the matrix of second partial derivatives. For the special case of isotropic materials, $Y^0 = Y^1 = 2(1+\nu)G^{01} = Y^{01}/\nu$.

In the following section, we show that orthotropic effects can be captured by weighting each hinge stiffness by a scalar factor, $\overline{\lambda}_i$. For convenience of implementation, the results of this derivation are summarized in §3.5.1.

3.4.1 Derivation of hinge stencil orthotropy

In §3.2.2 we discussed a simple model of discrete bending energy for isotropic shells. We based this model on summing the contributions of squared mean curvatures over interior edges. While discrete mean curvatures suffice to cover the isotropic case, we must use a discretization of the full shape operator in the anisotropic case. Indeed, smooth anisotropic energy density (3.9) requires expressing the discrete shape operator as a symmetric 2×2 matrix in the material frame—as the entries of this matrix measure bending and shearing stiffness in principal material directions. A hinge-based discrete shape operator $\mathbf{S}$ has been successfully applied in geometric modeling [Hildebrandt and Polthier, 2004]. Here we augment the geometric modeling view by two aspects: (a) we express $\mathbf{S}$ in a material frame as opposed to its original expression in a principal curvature frame, and (b) we show how $\mathbf{S}$ leads to a scalar stiffness-correcting factor per edge covering the discrete orthotropic case.

Let us briefly recall the geometric view based on discrete principal curvatures taken in [Hildebrandt and Polthier, 2004]. It is well-known from the smooth case [do Carmo, 1992] that $\mathbf{S}$ corresponds to a quadratic form whose (orthogonal) eigenvectors correspond to the two principal curvature directions, and its eigenvalues correspond to the principal curvatures, κ_0 and κ_1. Recall also that mean curvature is given as $H = \kappa_0 + \kappa_1$. In the discrete *edge-based* view, [Hildebrandt and

Polthier, 2004] defines the two principal curvature directions as (1) the direction along the edge and (2) the direction perpendicular to the edge—with no curvature ($\kappa_0 = 0$) along the edge and normal curvature ($\kappa_1 = \kappa$) perpendicular to the edge. Consequently, $H = \kappa$, and in the edge-based principal curvature frame, the discrete shape operator takes the form

$$\mathbf{S} = \begin{pmatrix} 0 & 0 \\ 0 & \kappa \end{pmatrix} = H \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} .$$

We note that this hinge-based shape operator is mesh dependent (in particular, principal directions are tied to edge directions). However, if the edge directions are distributed evenly, on the average the true shape operator is well approximated.

In order to express $\mathbf{S}$ in the material frame, we treat the mesh area associated with each hinge stencil as a *homogeneous* piece of material, *i.e.*, we assume that the material parameters cY^0, cY^1, cY^{01}, hG^{01}, and the material axes are *constant* over hinge stencils. Denoting by γ_0 the angle between edge $\mathbf{e}_i$ and the first material axis, $\hat{\mathbf{y}}_a$, and using the fact that the shape operator transforms like a quadratic form, we obtain that $\mathbf{S}$ takes the form

$$\mathbf{S}_i = H_i \begin{pmatrix} \sin^2\gamma_0 & -\sin\gamma_0\cos\gamma_0 \\ -\sin\gamma_0\cos\gamma_0 & \cos^2\gamma_0 \end{pmatrix} ,$$

when expressed in the material frame of the edge i. Finally, using the smooth energy density (3.9) and setting $\epsilon = \mathbf{S}$, it follows from basic trigonometric identities that the discrete orthotropic density can be written as

$$\underbrace{\left((cY^0)\sin^4\gamma_0 + (cY^1)\cos^4\gamma_0 + \frac{1}{2}\left(cY^{01} + hG^{01}\right)\sin^2(2\gamma_0) \right)}_{\overline{\lambda}_i} \cdot H_i^2 ,$$

providing a way to adjust hinge weights to effect orthotropic response.

This completes the requisite derivations for orthotropic cubic shells. In the following, we discuss the implementation of our model (§3.5–§3.6) and demonstrate the generality of the model with various simulation examples (§3.7).

3.5 Implementing cubic shells

In this section we describe the computation of forces and force Jacobians for the cubic shells energy, (3.7). Consider a mesh with n vertices and coordinate vectors $\mathbf{x}^x, \mathbf{x}^y, \mathbf{x}^z \in \mathbb{R}^n$. *Hinge-centric* computations are expressed in terms of the hinge's *local indices* (Fig. 3.1) for triangles, $\{T_0, T_1\}$, edge vectors, $\{\mathbf{e}_0, \ldots, \mathbf{e}_4\}$, and vertex positions, $\{\mathbf{x}_0, \ldots, \mathbf{x}_3\}$.

Precomputation. Recall that quantities that depend only on the undeformed positions are decorated with a bar, *e.g.*, $\overline{\mathbf{x}}$. Compute barred quantities once, before the simulation.

One-time matrix assembly. Assemble the global $n \times n$ matrix, $\overline{\mathbf{Q}}$, by iterating over hinge stencils. In the usual style of *stiffness matrix assembly* [Zienkiewicz and Taylor, 1989], $\overline{\mathbf{Q}}$ accumulates contributions from each *local* 4×4 matrix, $\overline{\mathbf{Q}}_i$:

$$\overline{\mathbf{Q}}_i = \left(\frac{3\overline{\lambda}_i \cos\overline{\theta}_i}{\overline{A}_i} \right) \overline{K}_i^T \, \overline{K}_i ,$$

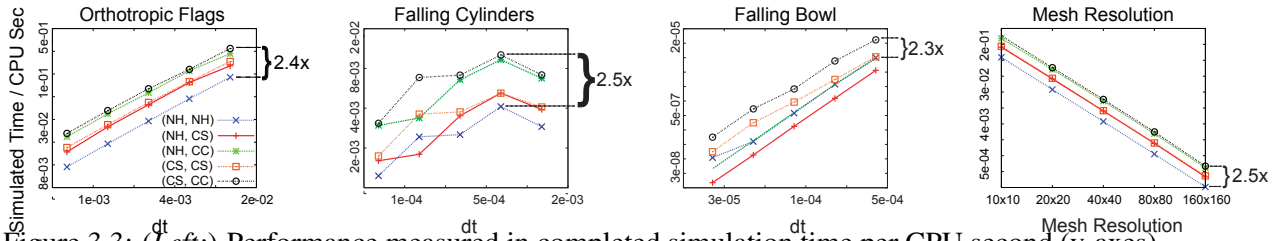


Figure 3.3: (*Left:*) Performance measured in completed simulation time per CPU second (y-axes) as a function of discrete time step (x-axes). Higher values mean better performance. In an inexact Newton framework, Cubic Shells dominate performance for all tested scenarios, including (left-to-right) Orthotropic Flags (Fig. 3.6), Falling Cylinders (Fig. 3.4), and Falling Bowl (Fig. 3.8). *Right:* Performance as a function of mesh resolution (from 10×10 to 160×160 regular grids), demonstrating that the benefits of cubic shells are not resolution-dependent. Timing numbers include timestepping and collision detection/response.

where $\overline{\lambda}_i$ is the stiffness of the hinge, $\overline{A}_i$ is the combined area of the two hinge triangles, and $\overline{\theta}_i$ is the undeformed hinge angle. In local indices, $\cos\overline{\theta} = -\langle \overline{\mathbf{t}}_0^{(0)}, \overline{\mathbf{t}}_0^{(1)} \rangle / |\overline{\mathbf{e}}_0|^2$, and $\overline{K} = (\overline{c}_{03} + \overline{c}_{04}, \overline{c}_{01} + \overline{c}_{02}, -\overline{c}_{01} - \overline{c}_{03}, -\overline{c}_{02} - \overline{c}_{04}) \in \mathbb{R}^4$, where $\overline{c}_{jk} = \cot \angle \overline{\mathbf{e}}_j, \overline{\mathbf{e}}_k$.

Force computation. Compute forces by adding thin plate and nonflat contributions. Compute thin plate contributions globally as the matrix-vector products $\mathbf{f}^x = \overline{\mathbf{Q}}\mathbf{x}^x$, $\mathbf{f}^y = \overline{\mathbf{Q}}\mathbf{x}^y$, $\mathbf{f}^z = \overline{\mathbf{Q}}\mathbf{x}^z$. Compute nonflat contributions by accumulating local hinge contributions

$$\begin{aligned} \mathbf{f}_0 &= -\mathbf{f}_1 - \mathbf{f}_2 - \mathbf{f}_3 \,, & \mathbf{f}_1 &= \overline{k}(\mathbf{e}_1 \times \mathbf{e}_2) \,, \\ \mathbf{f}_2 &= \overline{k}(\mathbf{e}_2 \times \mathbf{e}_0) \,, & \mathbf{f}_3 &= \overline{k}(\mathbf{e}_0 \times \mathbf{e}_1) \,, \end{aligned}$$

where $\overline{k} = 3\overline{\lambda}_i(\overline{c}_{01} - \overline{c}_{03})(\overline{c}_{04} - \overline{c}_{02})[\overline{\mathbf{e}}_0, \overline{\mathbf{e}}_1, \overline{\mathbf{e}}_2]/(\overline{A}_0|\overline{\mathbf{e}}_0|^3)$.

Force Jacobian computation. The exact force Jacobian ($\mathbb{R}^{3n} \times \mathbb{R}^{3n}$) is obtained by adding the thin plate and the nonflat contributions. The thin plate contributions are given by $\partial\mathbf{f}^x/\partial\mathbf{x}^x = \partial\mathbf{f}^y/\partial\mathbf{x}^y = \partial\mathbf{f}^z/\partial\mathbf{x}^z = \overline{\mathbf{Q}}$. For best performance (without sacrifice of accuracy), the nonflat contribution is omitted (as explained in §3.6); for completeness of exposition the nonflat contribution is detailed in the Appendix A.

3.5.1 Orthotropic hinge

The simple expressions above are all that is required to implement a cubic hinge-based bending force, given a stiffness value $\overline{\lambda}_i$ per hinge. For homogeneous isotropic materials, $\overline{\lambda}_i = \overline{\lambda}$ does not vary over the mesh, and the above derivation suffices; if a general class of orthotropic materials is desired, then $\overline{\lambda}_i$ should be computed by the formula derived in (§3.4.1); recall $\overline{\lambda}_i =$

$$(cY^0)\sin^4\gamma_0 + (cY^1)\cos^4\gamma_0 + \frac{1}{2}\left(cY^{01} + hG^{01}\right)\sin^2(2\gamma_0) \,,$$

where γ_0 is the angle between the hinge edge and the first material axis, $\hat{\mathbf{y}}_0$, and Y^0, Y^1, Y^{01}, and G^{01} are the material parameters described above. The above expression is all that is needed to implement orthotropic bending for cubic shells, and indeed for any existing hinge-based model.

In our implementation, the (spatially-varying) direction of $\hat{\mathbf{y}}_0$ is encoded by a coordinate function over a given parameterization of the surface. In particular, we use the color values of a texture map to encode the direction of the material axis.

3.6 Efficient simulation of thin shells

The separability of the force Jacobian of cubic shells into a constant and linear term opens several interesting possibilities for efficient time integration of thin shell dynamics. Among common discrete time-stepping schemes [Hauth, 2004; Hairer *et al.*, 2006; Baraff and Witkin, 1998], implicit schemes are popular in animation due to their stability. Implicit schemes advance time by solving a (typically nonlinear) system of equations. The system is usually solved by repeated Newton iterations (although *semi-implicit* methods complete a single Newton iteration) [Press *et al.*, 1992a]. Each Newton iteration requires an evaluation of the force, $-\nabla E$, as well as the force Jacobian.

The fixed points of Newton's method remain unaltered when the nonlinear system Jacobian is replaced by any invertible matrix. An inexact Newton's method [Morini, 1999] uses this fact in replacing a costly Jacobian with an inexpensive approximant. Hauth [Hauth, 2004] used an approximation of the in-plane stretching force Jacobian, and an inexact Newton framework, to accelerate cloth simulations; our work is similar, but we approximate the bending Jacobian. Wardetzky *et al.* [Wardetzky *et al.*,] also used an inexact Newton solve, in a mesh smoothing application; in contrast to that application, we treat the dynamics of shells having nonflat undeformed configurations.

Experiment. We consider the application of the inexact Newton's method with approximations to the bending force Jacobian. To provide a standard and easily-reproducible point of reference, we consider the Euler method, fully-implicit on stretching and bending elastic forces, without any additional damping forces.

In our experiments we pair two possible bending forces with three possible force Jacobians. For the two forces we consider the *nonlinear hinge* (NH)—the force resulting from treating bending energy fully nonlinearly (*i.e.*, by dropping the isometry assumption)—and the *cubic shells force* (CS) arising from our cubic energy as discussed in the previous section. For the three force Jacobians we consider the *nonlinear hinge Jacobian* (NH), the *entire* cubic shells Jacobian (CS), and the *constant part* (CC) of the cubic shells Jacobian. A particular choice of (inexact) Newton is denoted by a 2-tuple, *e.g.*, (CS,CC) denotes the use of the cubic shell bending forces paired with an approximate constant Jacobian; (NH,NH) denotes the usual fully-implicit implementation of the nonlinear hinge.

As a baseline, we run (NH,NH) at the maximum stable step size, and four smaller step sizes; these step sizes are reused in runs of (NH, CS), (NH, CC), (CS, CS), (CS, CC). We measure performance of these runs for various problem scenarios (see §3.7 and video).

We record the performance of each method, measured as the ratio *completed simulation time per CPU second*, *i.e.*, higher values mean better performance (see Figure 3.3–left). The cubic shells with an approximate *constant* Jacobian, (CS,CC), dominates all methods for all problem scenarios. Note that the performance gains observed for (CS, CC) are independent of mesh resolution (see Figure 3.3–right).

3.7 Visual impact of orthotropic parameters

Orthotropic parameters provide additional artistic control over the dynamics of wrinkles and folds. Consider for example a flag tailored by cutting a rectangular pattern from an orthotropic textile. The

orientation of warp and weft relative to the flag's pattern, as well as the values of Young's and shear moduli, have considerable visual impact on the resulting dynamics (see Fig. 3.6 and accompanying video).

The orthotropic shear modulus plays an important role in the *drapability* of a fabric [Sidabraite and Masteikaite, 2002]; the shear modulus is *not* captured by geometric models that employ Euler's formula, such as [Baraff and Witkin, 1998; Volino and Magnenat-Thalmann, 2006b], or elliptic interpolation, such as [Choi and Ko, 2003]. A high shear modulus tends to align folds and wrinkles to the material axes; a lower shear modulus allows the fabric to fold along other directions, and therefore to obtain a closer fit to the body. The first two frames of Fig. 3.5 compare a high and low shear modulus, respectively, illustrating the intuitive notion of drapability. When a high shear modulus is desired (leftmost frame), the Young's modulus is a poor substitute—it is unable to capture stiff extrusion of the poncho around the shoulders without introducing extraneous stiffness elsewhere.

Furthermore, orthotropy has a strong effect on the interaction between a material and its environment. We simulate the fall and bounce of four elastic cylinders, each with a different orientation for its principal material axis. The resulting animation (Fig. 3.4 and accompanying video) reveals the extreme variations in deformation and overall trajectory that arise purely from changing the orthotropic parameters.

Orthotropy indirectly enriches any other technologies implemented in the simulator, such as viscoelasticity or fracture [Terzopoulos and Fleischer, 1988]. For example, the anisotropic bending resulting from collisions produces distinctive fracture patterns both for plates (Fig. 3.7) and shells (Fig. 3.8). For a summary of the fracture algorithm refer to Appendix B.

3.8 Discussion

Limitations. The cubic hinge does not overcome those limitations that are shared by all hinge-based approaches. In particular, the simplicity of hinge-based models comes at the cost of limited convergence behavior and meshing-dependence [Grinspun *et al.*, 2006].

Implicit to the cubic hinge is the assumption that the surface deforms isometrically, *i.e.*, without stretching. A natural question, then, is how much stretching is permissible in practice, and what are the failure modes of the model under excessive stretching? To explore these questions, we re-simulate the falling cylinder and falling bowl using progressively lower stretching resistance. Both examples remain well-behaved so long as the stretching stiffness is two orders of magnitude greater than the bending stiffness; during the simulation, the meshes stretch by as much as 15%.

If we reduce stretching stiffness even further, then for the thin *shell* examples we observe a slow, noticeable stretching of the surface, in particular, in an expanding (not oscillatory) mode. This is perhaps not surprising, since a cubic energy is not bounded from below; under normal (quasi-isometric) circumstances, the stretching resistance prevents this infinite well from being exploited.

We repeat the above experiment with the billowing flag. Since the flag has a flat undeformed configuration, the cubic energy term vanishes, leaving a quadratic energy bounded from below. We observe that the flag simulation is well-behaved even when strong wind induces stretching of 300%.

Conserved momenta. In all circumstances (even when parasitic stretching is observed due to a broken isometry assumption), the simulated model conserves linear and angular momenta. To see this, note that the quadratic forces arising from the cubic energy are *not* approximated (only their Jacobian is); since the cubic energy is invariant under rigid body transformations of the deformed (and

also the undeformed) configuration, forces induced by this energy do not apply a global acceleration or torque.

Due to its simplicity, efficiency, and generality, we expect cubic shells to be of practical relevance in computer animation.

3.9 Appendix A: Neglected term in force Jacobian

As discussed in §3.6, we advocate the use of an inexact Newton's method, using only the constant portion of the force Jacobian given in §3.5. Indeed, for optimal performance (and no loss of accuracy) one should omit calculation of the linear component arising from the nonflat undeformed configuration. However, for completeness of presentation, we include the expression for this linear component below.

With respect to the local position vector $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \in \mathbb{R}^{12}$, and the corresponding local force vector $\mathbf{f} = (\mathbf{f}_0, \mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3) \in \mathbb{R}^{12}$, the local force Jacobian ($\mathbb{R}^{12} \times \mathbb{R}^{12}$) is given by

$$\overline{k} \begin{pmatrix} 0 & (\mathbf{e}_1 - \mathbf{e}_2)^* & (\mathbf{e}_2 - \mathbf{e}_0)^* & (\mathbf{e}_0 - \mathbf{e}_1)^* \\ -(\mathbf{e}_1 - \mathbf{e}_2)^* & 0 & -\mathbf{e}_2^* & \mathbf{e}_1^* \\ -(\mathbf{e}_2 - \mathbf{e}_0)^* & \mathbf{e}_2^* & 0 & -\mathbf{e}_0^* \\ -(\mathbf{e}_0 - \mathbf{e}_1)^* & -\mathbf{e}_1^* & \mathbf{e}_0^* & 0 \end{pmatrix},$$

where each entry represents a 3×3 skew-symmetric subblock; the asterisk applied to a vector, $\mathbf{v}^*$, produces the matrix

$$\mathbf{v}^* = \begin{pmatrix} 0 & -\mathbf{v}_z & \mathbf{v}_y \\ \mathbf{v}_z & 0 & -\mathbf{v}_x \\ -\mathbf{v}_y & \mathbf{v}_x & 0 \end{pmatrix},$$

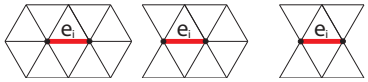
corresponding to the cross product operation $\mathbf{v} \times (\cdot)$. The local force Jacobian is assembled into the global $\mathbb{R}^{3n} \times \mathbb{R}^{3n}$ Jacobian in the usual manner. While it is most easily expressed by a skew-symmetric matrix of skew-symmetric subblocks, note that $\operatorname{Hess} G(\mathbf{e}_0)$ is indeed symmetric.

3.10 Appendix B: Implementing fracture

The material fractures when internal strain exceeds a threshold. For simplicity we only consider fracture along existing mesh edges; however, this limitation can be easily removed, see *e.g.*, [O'Brien and Hodgins, 1999; Molino *et al.*, 2004; Gingold *et al.*, 2004]. A hinge edge, $\mathbf{e}_i$, fractures if the bending strain, $\frac{|\mathbf{e}_i|}{\overline{A}_i}(\theta_i - \overline{\theta}_i)$, exceeds a material-dependent threshold.

Fracture may be viewed as the transition when an interior edge becomes a boundary edge. A mesh edge has one of three states: INTERIOR→FRACTURED-INTERIOR→BOUNDARY, where the arrows indicate allowable state transitions. An INTERIOR edge becomes FRACTURED-INTERIOR if the strain threshold is exceeded; a FRACTURED-INTERIOR edge becomes BOUNDARY only as a consequence of explicit changes to mesh connectivity, as explained below.

A BOUNDARY vertex is a vertex incident on a FRACTURED-INTERIOR or BOUNDARY edge. A FRACTURED-INTERIOR edge, e_i, may be in one of three configurations, distinguished by the number of incident BOUNDARY vertices:



B0 **B1** **B2**

In each case, any BOUNDARY vertices incident on e_i are split into two. Specifically, in case: (**B0**), no action is taken; (**B1**) and (**B2**), one and two vertices are split, respectively, and the resulting change to mesh connectivity causes at least one edge (e_i) but possibly other incident edges to transition FRACTURED-INTERIOR→BOUNDARY.

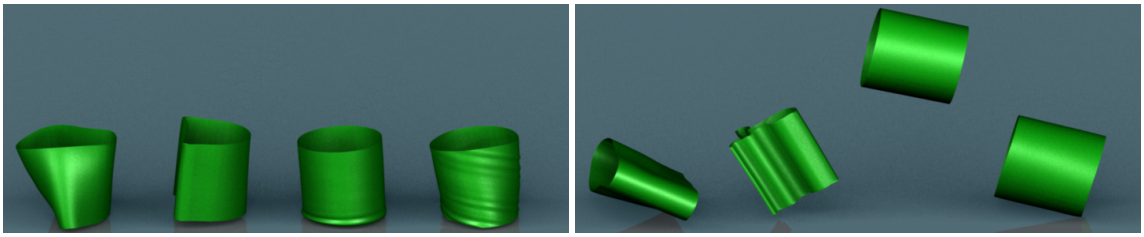


Figure 3.4: Thin-shell simulation of falling orthotropic cylinders. Material axes vary from left-to-right: isotropic, vertical, horizontal, diagonal. Left Image: stress formed on the cylinders upon impact. Right Image: the effects of anisotropy after impact. Note the extreme flatenning of the cylinder with axis aligned vertically and the high bounce of the cylinder with axis aligned horizontally.

$Y^0 = Y^1 = 0.01, G^{01} = 100$ $Y^0 = Y^1 = 0.01, G^{01} = 0$ $Y = 0.2$ $Y = 0.35$ $Y = 0.5$

Figure 3.5: Shear vs. no bending shear: a high bending shear modulus tends to align folds and wrinkles to the material axes (leftmost); a lower shear modulus allows the fabric to fold along other directions, and therefore to obtain a closer fit to the body (second frame). Shearing effects cannot be reproduced by simply tuning Young moduli (3 unsuccessful attempts shown on the right).

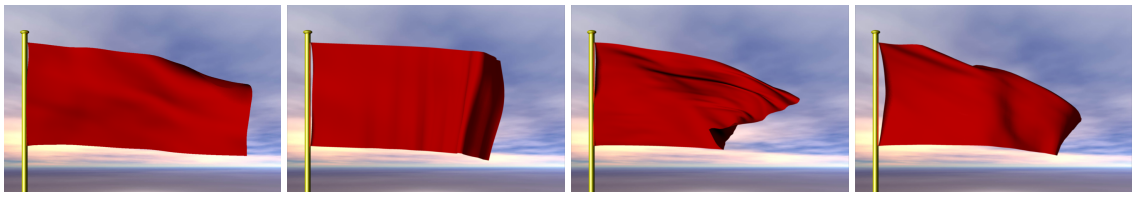


Figure 3.6: Varying material axes at no additional cost. From left-to-right: isotropic, vertical, horizontal and diagonal.

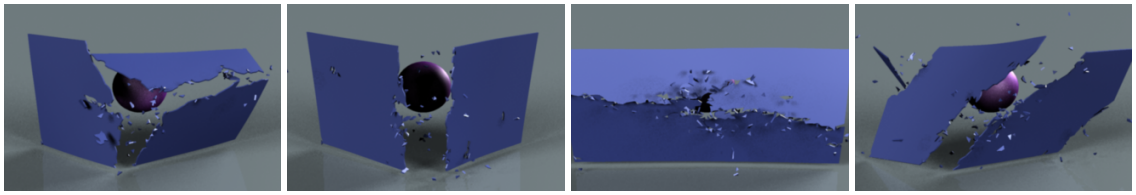


Figure 3.7: Thin plates: adjustable directions of fracture patterns. Left-to-right: isotropic, vertical, horizontal and diagonal.

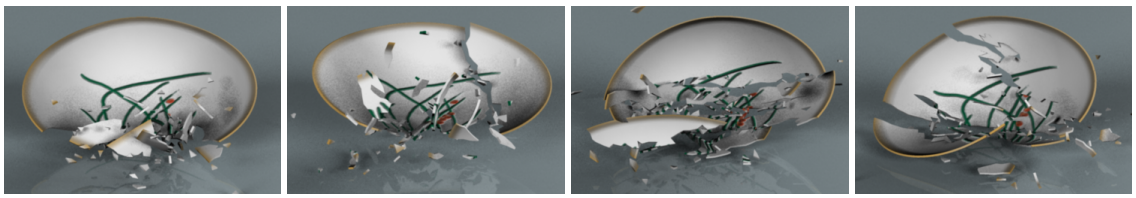


Figure 3.8: Thin shells: adjustable directions of fracture patterns. Left-to-right: isotropic, vertical, horizontal and diagonal.

Chapter 4

Wire Mesh Design

4.1 Introduction

Wire meshes enjoy broad application in art, architecture, and engineering, including handmade sculptures, filters, support structures in composite materials, and architectural faccades (see Fig. 4.2). Despite their widespread use, a systematic design methodology for freeform wire meshes is lacking. While physical exploration helps build intuition in early concept design, rationalizing a surface entails numerous constraints that are often globally coupled. Artists currently use an incremental, trial-and-error approach, where an initially flat piece of wire mesh is gradually bent by hand to conform to a desired surface. Likewise, in architecture wire meshes are restricted to very simple shapes, such as planes, cylinders, cones, or half-spheres, despite great interest in freeform faccades. We show that a much richer space of wire meshes can be more effectively explored using digital tools, which automatically account for the strong global coupling of physical and geometric constraints.

While in our fabrication examples (but not for our design tool), we have focused on wire mesh made of steel, wire mesh encompasses a much broader range of materials, such as fishnet stockings, woven reinforcements in composite materials, or even onion nets. Indeed, even something as prosaic as a simple onion net reveals some of the core structural properties of wire mesh: inextensible fibers that are woven in a criss-cross pattern such that the warp and weft directions cannot stretch but may significantly shear towards (or away from) one another (see Fig. 4.4). In order to gain intuition for designing with wire mesh, one may try to "dress" a given target shape, such as a vase, a bust, or a ball with an onion net. Soon one then discovers that due to shearing some features cannot be captured, that more material may be required in certain areas, or that it is difficult to preserve the fine details of the given target shape.

Such difficulties are ubiquitous when working and designing with wire mesh: If a wire mesh is required to lie exactly on a given target design surface, incremental layout often fails to adequately represent the desired shape. We substantiate this observation by modeling wire meshes as discrete Chebyshev nets (§4.3), revealing fundamental limitations in the kind of shapes that can be equipped with a single wire mesh. Further insights from the theory of Chebyshev nets allow us to formulate an optimization scheme where the mesh can deviate from the target surface in a controlled way. This scheme balances inextensibility of wires and limits on shearing angles with design objectives imposed by the user. Optimization is then incorporated into an interactive design tool that leverages the user's high-level understanding of shape (§4.4). The tool interleaves user input on artistic decisions with global optimization to explore the design space. This facilitates an effective pipeline from surface

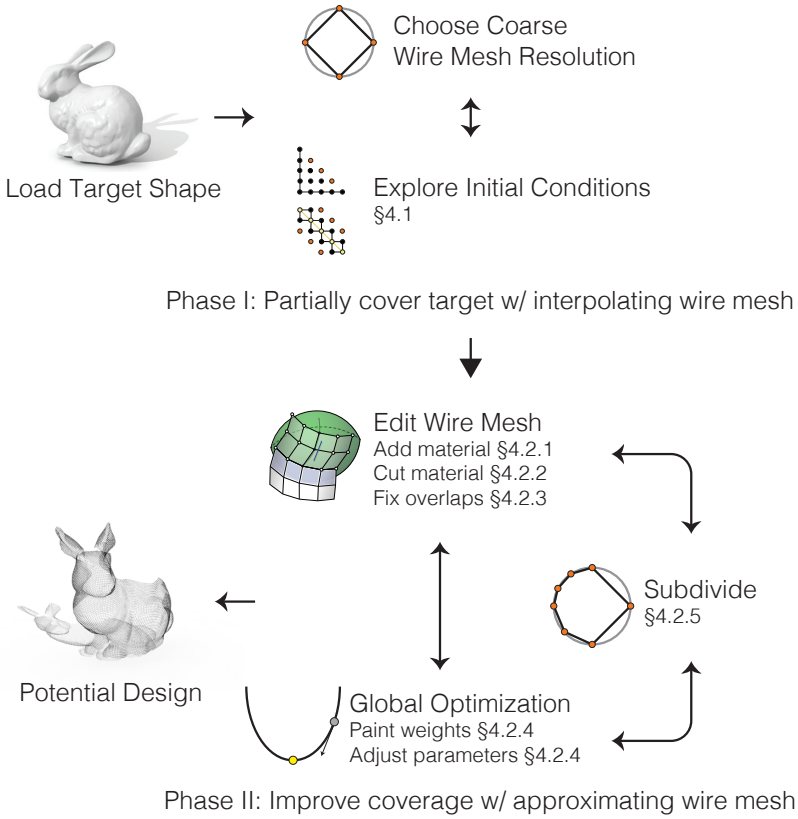


Figure 4.1: The wire mesh design process.

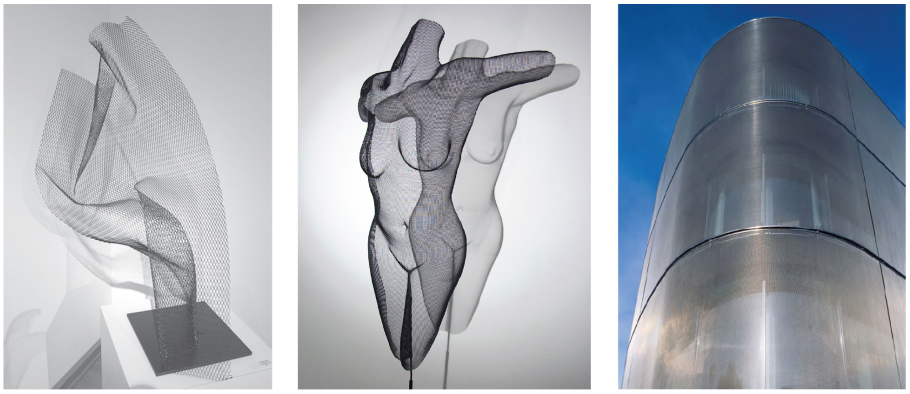
modeling to physical realization.

In contrast to previous approaches, our methodology (i) lifts the restriction of a priori curvature bounds for a given target shape (which is omnipresent in the mathematical literature and previous computer-aided tools) and (ii) works without insertion of darts (i.e., folds stitched into the material). Indeed, unlike garments, where the inclusion of darts are considered a premium and a signature of thoughtful design, darts generally introduce a point of failure, over-engineering, or manufacturing complexity in wire mesh structures. Instead, our design tool allows the user to interactively and iteratively grow a wire mesh on a target shape, thus allowing fabrication with a *single piece of material*.

Our contribution is an example of how geometric modeling and optimization-based shape exploration can lead to new material-aware design solutions that enable creative works not feasible before. The design process (Fig. 4.1) has two phases: a setup phase that directly extends previous work on *interpolating* Chebyshev nets which partially cover the target and a novel design loop where the designer interactively explores *approximating* Chebyshev nets to increase coverage.

4.2 Related Work

Computational tools for material- and fabrication-aware design have recently become a prominent topic in computer graphics research. A typical example are algorithms for the design and optimization



© Donald Kolberg © Randy Cooper wiremesh facade

Figure 4.2: Wire meshes are a popular medium in abstract (left) and figurative (middle) sculptural art. These freeform shapes are created by manually bending a single, flat piece in an incremental, trial-and-error process. Lacking an effective digital design process, the use of wire meshes in architectural faccades (right) is currently limited to simple geometries.

of discrete freeform surfaces with planar polygons [Liu *et al.*, 2006; Wang *et al.*, 2008; Bouaziz *et al.*, 2012; Poranne *et al.*, 2013]. Ensuring planarity of mesh elements facilitates the use of cost-effective materials and construction technologies, for example in stone or glass faccades. Additional constraints such as torsion-free nodes can be incorporated to improve structural performance and further simplify the fabrication process [Liu *et al.*, 2006]. Related examples include rationalization and shape exploration for developable surfaces [Pottmann *et al.*, 2008], geodesic patterns [Pottmann *et al.*, 2010], curved panelings [Eigensatz *et al.*, 2010], or circular arc structures [Bo *et al.*, 2011]. A series of papers exploits geometric abstractions of compression-only surfaces to facilitate the design of self-supporting structures [Vouga *et al.*, 2012; Panozzo *et al.*, 2013; de Goes *et al.*, 2013; Liu *et al.*, 2013]. The construction of planar intersecting pieces has been investigated by Schwartzburg and Pauly [Schwartzburg and Pauly, 2013] who map assembly and fabrication restrictions to geometric constraints of a mixed discrete/continuous optimization. The common theme of these and other related methods is that material behavior or physical restrictions are mapped to *geometric properties* or constraints of the design surface—a methodology that we also follow here. This strategy avoids the complexities of a full physical simulation and enables efficient computations for interactive form-finding and design exploration.

As explained in §4.3, wire meshes are best modeled by Chebyshev nets—a geometric model of woven materials using a two dimensional net composed of inextensible yarns, first proposed by Chebyshev in 1878 [Tschebyscheff, 1878]. The same model is used in the mechanical theory of pure networks, i.e., grids of inextensible yarns with no shear resistance [Rivlin, 1958; Rivlin, 1964; Rivlin, 1997]. When shearing is incorporated, the model is known as a reinforced network [Adkins, 1956]; investigations into bending resistant inextensible networks have been considered in [Wang and Pipkin, 1986]. Pipkin analyzed stress in reinforced networks on arbitrary curved surfaces [Pipkin, 1984] and the distribution of wrinkles of the solution [Pipkin, 1986], but assumes—different from our approach—that the reinforced network already lies on the surface. A purely discrete version of Chebyshev nets has been used in computer aided design to model the forming of woven reinforced composite materials to surfaces. However, many studies focus only on simple geometries such

as a hemisphere [Robertson *et al.*, 1981; Ye and Daghyani, 1997], polyhedral and rounded cones [Robertson *et al.*, 2000; Baillargeon and Vu-Khanh, 2001] or translated sine curves [Wang *et al.*, 1999].

In terms of application domain, virtual design with Chebyshev nets was considered by Aono et al. [Aono, 1994; Aono *et al.*, 1996; Aono *et al.*, 2001]. Building upon initial computational investigations [van West *et al.*, 1990], Aono et al. presented a method for finding Chebyshev nets *interpolating* a given surface via automatic dart insertions. Their approach mimics the process by which a tailor drapes garments over the human form, pinning initial lines of material onto a dress form, then working outwards from these constraints, making cuts or inserting darts as required to fit the underlying form. Our work is inspired by and builds upon the work of Aono and coworkers, while (i) lifting the restriction that total Gaussian curvature be bounded by 2π (thereby expanding the scope of possible designs), (ii) working with a *single* piece of material without the need of inserting darts, (iii) expanding the range of admissible target shapes (including cylindrical topologies), and (iv) extending modality by deeply integrating the user into the design loop with a varied tool set.

4.3 Chebyshev Nets

As prototypical wire meshes we consider metal (steel) wires woven in a *plain weave*. In these most ubiquitous wire meshes, longitudinal *warp* and transversal *weft* wires are interwoven (but not welded) in a criss-cross pattern (see inset figure). For the typical forces and deformations applied to wire meshes, the stretching of each metal wire may be reasonably neglected, thus each wire is adequately modeled as an inextensible elastic curve. The elastic response of the ensemble is then governed by the bending of each curve and the interactions induced by the weave pattern.

The weave induces a "soft" interlocking of wire: while each wire may slightly slide over the crossing wires, significant sliding is uncharacteristic because it occurs only under exceedingly large forces. Consequently, adjacent contact points maintain their intrinsic distance, even for large extrinsic deformations, resulting in a structure where warp and weft directions cannot stretch but significantly shear towards or away from one another. The corresponding mathematical model is the theory of *Chebyshev nets*.

Chebyshev nets are akin to the conformal parameterizations commonly used for texture mapping. While conformal maps exactly preserve *angles* but allow for uniform stretching, Chebyshev nets preserve *lengths* along two parameter (warp and weft) directions but allow for shearing of angles between warp and weft. Let $r : \mathbb{R}^2 \to \mathbb{R}^3$ be a parameterization of a smooth surface describing a patch of a surface in space. Then $r(u, v)$ is *Chebyshev* if $|r_u| = |r_v| = 1$, where (u, v) is an orthonormal coordinate system for $\mathbb{R}^2$. A collection of such patches describing a whole surface, such that coordinate transitions are given by *translations* only, is called a *smooth Chebyshev net*.

4.3.1 Smooth Chebyshev Net Theory

One of the intricate mathematical difficulties for constructing smooth Chebyshev nets results from the fact that while one can *locally* equip every smooth surface in 3-space with a Chebyshev net [Bieberbach, 1926], this is no longer the case *globally* without producing singularities. This mathematical difficulty translates into very concrete challenges in the design process. While it is locally possible to fit a wire mesh to a given shape, there are strong global obstructions; moreover, small local

changes might have drastic global effects—making manual design cumbersome, time consuming, or intractable.

Curvature and shear The coupling between shearing of the wire mesh and curvature makes *global* existence of Chebyshev nets on an arbitrary smooth surface a delicate matter. Let $\gamma(u, v)$ be the *shear angle* of a Chebyshev net $r(u, v)$, i.e., $\sin \gamma(u, v) = r_u(u, v) \cdot r_v(u, v)$. Simply, γ measures the signed angle deviation of the originally orthogonal warp and weft directions under the deformation of the surface. The so-called Gauß equation for the local parameterization reads [Pipkin, 1984]

$$\mathcal{K}(u, v) \cos \gamma(u, v) = \gamma_{uv}(u, v) \, , \tag{4.1}$$

where $\mathcal{K}(u, v)$ denotes the Gaussian curvature at the point $r(u, v)$. This equation reveals that changes in the shearing angle directly correspond to the encoding of curvature. Indeed, regions where the magnitude of Gaussian curvature is high correspond to large magnitude (close to $\pi/2$) shearing angles or a high rate of change of shearing angles (which then leads to large magnitude shearing angles nearby). For wire mesh design, where large magnitude shear angles are prohibitive (see Fig. 4.4), this results in difficulties for covering regions of high curvature.

Global obstructions to existence An important obstruction that results from the coupling between shear angle and curvature is provided by the *formula of Hazzidakis [Hazzidakis, 1879]*: Consider an axis-aligned (with respect to the u and v parameter directions) rectangular domain $D \subset \mathbb{R}^2$, then it follows from Equation (4.1) that

$$\int_D \mathcal{K}(u, v) \mathrm{d}A = 2\pi - \sum_{i=0}^{3} \alpha_i \, , \tag{4.2}$$

where $\mathrm{d}A$ is the area element on the surface and the α_i are the interior angles of the quadrangle given by the image of the axis-aligned rectangle D under the Chebyshev net r.

As a consequence of Hazzidakis' formula, *if* one requires the boundary of a rectangular patch D to coincide with parameter lines, then it is impossible to cover the image of D with a Chebyshev net if this image has total Gauß curvature greater than 2π. Perhaps surprisingly, despite the Hazzidakis obstruction, Voss [Voss, 1882] showed that there exists a global Chebyshev net on any surface of revolution which does not meet the rotation axis—even if total Gauß curvature exceeds 2π. Recently, Ghys [Ghys, 2011] proposed a Chebyshev net on the sphere (with singularities along two spherical arc segments at the south pole) that is different from the solution of Voss (with singularities at the poles where the profile curve meets the rotation axis). These results show that Hazzidakis' obstruction ceases to be valid *if one gives up on insisting on axis-aligned parameter domains*.

Inspired by these observations, our design tool introduced in §4.4, allows the user to both: construct *non axis-aligned domains* and to add or remove material, thus *changing the shape of the domain*.

Sufficient conditions for existence Hazzidakis' formula provides *necessary* conditions for the existence of Chebyshev nets. The search for optimal results about *sufficient* conditions for the existence of global Chebyshev nets on surfaces is still ongoing [Bakelman, 1965; Samelson, 1991; Samelson and Dayawansa, 1995; Burago *et al.*, 2007]. Although some of these works offer constructive proofs, they do not immediately lead to a computationally feasible algorithm. More importantly,

these works assume that total negative and positive Gauß curvature does not exceed 2π in magnitude—a bound too restrictive for real-world design—and that the resulting Chebyshev net lies *exactly* on the given surface—a requirement that is neither practical nor strictly necessary in design.

Summary The above mathematical properties translate into practical difficulties when designing shapes with wire mesh. We tackle these challenges in our design tool by (i) allowing the parameter domain to be changed by the user by adding or removing material, (ii) working with Chebyshev nets *nearby* (but not exactly on) a given target shape, and (iii) accommodating for the global nature of the problem with the help of an optimizer.

4.3.2 Discrete Chebyshev Nets and the Role of Shear

We model wire mesh by *discrete Chebyshev nets*, rhombic nets comprised of inextensible equilateral edges such that each interior mesh vertex has valence four. Notice that we do *not* require the rhombi to be planar. Akin to the smooth case, discrete Chebyshev nets have a long history in mathematics. Originally introduced for the special case of constant Gauß curvature surfaces [Wunderlich, 1951; Sauer, 1970; Bobenko and Pinkall, 1996; Hoffmann, 1999; Pinkall, 2008], their more general theory is still an active area of research.

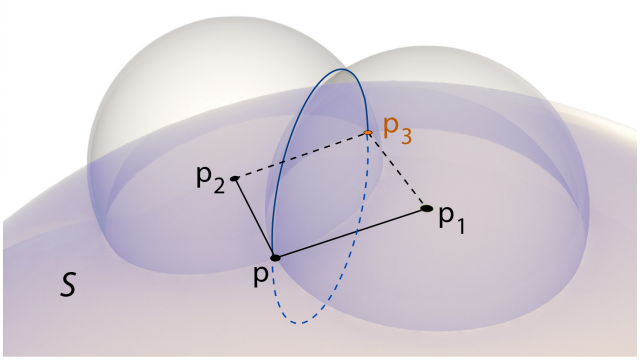
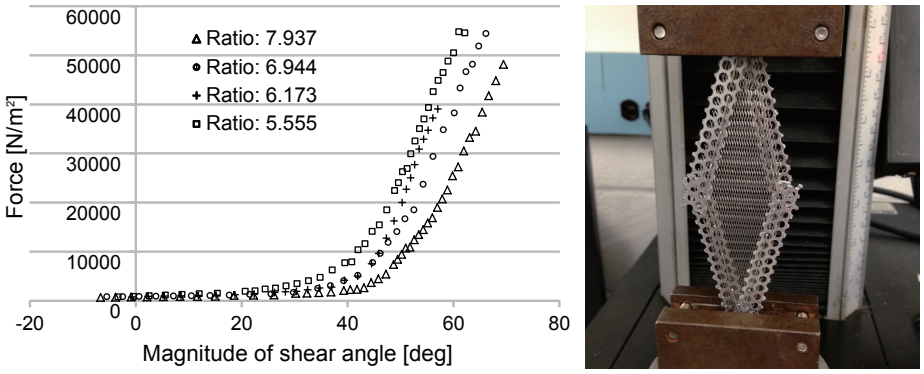


Figure 4.3: Local construction of a discrete Chebyshev net.

As in the smooth case, there exists a discrete Chebyshev net locally around any point p on a target surface $S \subset \mathbb{R}^3$. Indeed, given a small neighborhood U of p on S, choose an edge length ℓ and two unit vectors $v, w \in \mathbb{R}^3$ such that $p_1 = p + \ell v$ and $p_2 = p + \ell w$ are in U. Generically the three points p, p_1, p_2 determine a unique fourth point $p_3 \in S$ such that the quadrilateral (p, p_1, p_2, p_3) is a (non-planar) rhombus (see Fig. 4.3). To obtain p_3, consider the two spheres of radius l around p_1 and p_2, respectively, which intersect in a circle (shown in blue). Both p and p_3 lie on the intersection of this circle with S. Generically, p_3 is unique and distinct from p.

The role of shear While the wires of a wire mesh do not stretch and offer some resistance to bending of warp or weft directions, the quadrilateral structure of the weave allows for a considerable amount of *shear*, which is ultimately responsible for the rich set of possible wire mesh deformations. We measure the discrete signed shear angle, γ, as the deviation from 90° of the interior angles of the quadrilaterals. Wire meshes exhibit little resistance to in-plane shearing as long as the magnitude of



Force vs. magnitude of shear angle Instron measurement setup

Figure 4.4: Measurements of shear resistance for four different wire meshes. While very little force is required to shear the mesh initially, an exponential increase in force can be observed starting at a shear angle magnitude of about $45°$. In the measurements shown above, we have tested four wire meshes with different mesh opening / wire thickness ratios. The diameter of the wires in all meshes is 0.009 mm and the meshes have per inch 14 cells (ratio: 7.937), 16 cells (ratio: 6.944), 18 (ratio: 6.173) cells and 20 cells (ratio: 5.555), respectively.

the shear between warp and weft lines does not exceed a certain threshold. Beyond this point the required shear force increases drastically.

This claim is validated by several experiments that we conducted on real materials using an *Instron* machine, a device that measures tensile (or compressive) forces under a prescribed deformation. As shown in Fig. 4.4, the energetic cost of shearing wire mesh samples of varying gauge is negligible for shears with magnitude below a (consistent) threshold of approximately $45°$. To deform a wire mesh beyond this threshold requires excessive force.

These experiments validate a *bounded-shear* model. When optimizing for a discrete Chebyshev net, we thus restrict the magnitude of the allowable shearing to a user-defined bound; our examples set the *shear limit* or *shear bound* to $\gamma_{max} = \pi/4$. Each rhombus is thus individually restricted to interior angles between $[\pi/2 - \gamma_{max}, \pi/2 + \gamma_{max}]$. We point out that the motivation for our specific choice of γ_{max} arises from experiments—other choices are indeed possible. We use the adjective *realizable* to refer to *discrete, bounded-shear* Chebyshev nets.

4.3.3 Building Discrete Chebyshev Nets

A prevalent way for constructing discrete Chebyshev nets is through a process called *integration* from appropriate *initial data* or *initial conditions*. This approach plays a central role in our implementation for *initializing* a discrete Chebyshev net on a target surface.

Interpolating integration The observation that three points of a rhombus in a discrete Chebyshev net on a (smooth or triangulated) surface S determine the fourth point leads to a construction of Chebyshev nets from certain *initial condition* curves. To this end, consider a curve on S that is equidistantly sampled (with respect to the extrinsic metric of $\mathbb{R}^3$). We refer to such a curve as *Cauchy* initial data. Any vertex of this curve whose adjacent edges (with respect to its adjacent sample points) form an angle that obeys the shear limit constraint can be used as a seed for integration, see Fig. 4.5

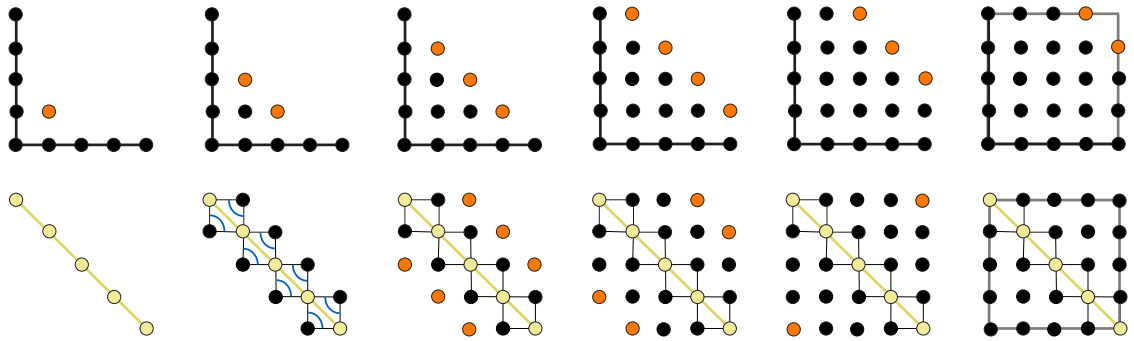


Figure 4.5: *Top:* integration using Cauchy initial data (black polygonal curve). *Bottom:* diagonal initial data given by a curve (yellow dots) and desired angles (blue, at black dots in 2nd picture from left); diagonal data determine two zig-zag curves along the diagonal. *Left-to-right:* orange dots denote new data that are computed from previously known ones. *Rightmost figure:* bounding box shows region in the parameter domain that can be covered.

(top) for a schematic illustration of this process. We additionally allow for specifying what we call *diagonal* initial data, which, different from Cauchy data, are given by a (discrete) curve on S together with angles (obeying the shear limit) for each curve segment. In this case, curve segments serve as diagonals of rhombi and angles specify the two (necessarily equal) angles opposite to the diagonal in each rhombus, see Fig. 4.5 (bottom). Notice that on a topological cylinder, diagonal initial data are *required* since they allow for covering the entire cylinder, while Cauchy data do not—see Fig. 4.6 for a schematic illustration.

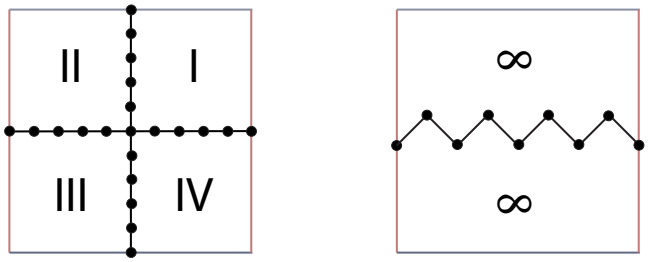


Figure 4.6: Influence of initial data for a topological cylinder—in both figures, left and right side of the square are identified (glued) to form a cylinder. *Left:* axis-aligned Cauchy data allow for covering a finite region only, i.e., the depicted parts of quadrants I to IV; additionally, in general there is no guarantee that the results of integration match on left and right—possibly leading to discontinuities. *Right:* zig-zag (from diagonal data) allows for covering an infinite cylinder—without further restrictions except for the shear limit.

Limits to interpolating integration While there exists a discrete Chebyshev net of some resolution ℓ around every point p on S given by initial data, it is impossible to know in advance how far these data will propagate the net. This makes it difficult to know if particular initial data are sufficient to construct a global discrete Chebyshev net. There are three ways in which initial data can fail

to propagate a realizable Chebyshev net to globally cover the entire target surface: (i) the generic construction of Fig. 4.3 fails to find a new point p_3 because the circle of intersection of the constraint spheres only meets the surface at the original point p; (ii) the three points p, p_1, p_2 already determine an angle that violates the shear limit; (iii) the prescribed initial data were not sufficient to cover the target surface because shearing in one place pulls material from another part, so more material is actually required. These three failure modes are dependent on the initial data and on the initial resolution, ℓ, at which the Chebyshev net was formed. Choosing ℓ too large in comparison to the target surface might produce an extreme approximation such as a single large rhombus for the entire target surface, while choosing ℓ too small might introduce unnecessary curvature detail that is either not of artistic interest or stems from artifacts of a triangulation.

Translation surface integration We employ an alternative method to construct discrete Chebyshev nets when interpolating integration fails. This translation surface integration method relies on the fact that three non collinear points p, p_1, p_2 in 3-space such that $|p - p_1| = |p - p_2|$ uniquely determine a fourth point p_3 such that (p, p_1, p_2, p_3) is a *planar* rhombus. That is, instead of propagating a Chebyshev net such that the fourth vertex resides on the target shape, we offer the possibility to propagate, from an initial curve on S, such that the resulting rhombi are *planar*, while maintaining the general integration paradigm depicted in Fig. 4.5 (top). Notice that in general the resulting net will deviate from the target shape—a property that is desirable for initializing an approximate (instead of interpolating) Chebyshev net in scenarios where integration fails. We refer to §4.4 for details of when we use translation surfaces instead of interpolating integration. We remark that discrete Chebyshev nets that are entirely comprised of planar rhombi are *discrete translation surfaces*—in analogy to smooth translation surfaces that are defined by $r(u, v) := a(u) + b(v)$, where $a, b : \mathbb{R} \to \mathbb{R}^3$ are smooth curves [Voss, 1882; Pipkin, 1984].

Next, we describe how these insights guide our design of wire mesh in practice.

4.4 Design Tool

We facilitate the creation of a *single, contiguous piece* of wire mesh that can be *cut out of the plane*, and bent *without inserting darts* and with *bounded shear* to approximate a desired surface, or *guide form*.

Theory informs us that the Chebyshev net constraints are globally coupled; when we additionally constrain the net to interpolate a given guide form, the design space is intractably small. Fortunately, considerable additional freedom can be gained by allowing for slight deviations from the guide form. We therefore seek designs that *approximate* rather than interpolate a given shape.

With a guide form given, the first phase is to create initial wire mesh material that *interpolates* a part of the guide form. Due to the limitations of interpolating integration laid out in the previous section, to extend coverage a second *approximating* phase is required. This second phase interweaves adding or removing wire mesh material, weight-painting, and globally optimization.

Coarse-to-fine design We find that a coarse-to-fine design process is effective. The designer first situates the wire mesh and resolves the coarsest features, before refining to focus on details. The design session begins by establishing a coarse (large cell size) wire mesh, and proceeds by iteratively

subdividing the wire mesh, revising the shape, and repeating, until the finest details are resolved. The revisions employ several types of tools, categorized as either *local* or *global* in effect.

Local vs. global tools Tools with local effect alter only the selected region of the wire mesh, e.g., adding or removing mesh material; these tools do not allow the *deforming* of the wire mesh: the Chebyshev constraints are inherently global in nature, prohibiting such a local deformation. To deform the mesh, we employ a *global* optimizer; it necessarily alters almost the entire wire mesh shape. This optimizer is incorporated into the interactive workflow, and the user controls the optimization by painting weights to prioritize approximation of some target regions over others.

We now survey these tools and refer the reader to the accompanying demonstrations in the supplementary video.

4.4.1 Phase I - Interpolating the Guide Form

We present two interpolating integration tools to quickly and easily lay out the initial patch of material. The first is a novel zig-zag tool for both cylindrical and disk topologies, while the second is the Cauchy integration tool for disk topology similar to the work of Aono et al. [Aono, 1994; Aono *et al.*, 1996; Aono *et al.*, 2001].

Zig-zag tool The designer draws a single "diagonal" curve segment on the surface, and specifies the (possibly varying) shear along the diagonal; using these data, the computer generates two curves that zig-zag on and off the diagonal, one on each side of the diagonal (see Fig. 4.5-bottom); the computer then integrates an interpolating wire mesh patch outward, using the diagonal initial data. To specify *cylindrical topology* a closed loop on the surface is specified as the initial diagonal curve instead of a curve segment.

Cauchy tool The designer draws two curve segments that *intersect* at a point; the computer then integrates an interpolating wire mesh patch from each quadrant of Cauchy initial data (see Fig. 4.5-top). Using the Cauchy tool with a geodesic curve, for instance, ensures a constant shear along the geodesic. Indeed, the geodesic curvature of the u-parameter lines and v-parameter lines are given by γ_u and $-\gamma_v$, respectively [Pipkin, 1984]. Therefore, a Chebyshev net's parameter line lies along a geodesic if and only if the shearing γ is constant along that line. Using geodesics as initial conditions makes the Cauchy approach appealing if a geodesic is made to pass over multiple hills and valleys. We found this useful when designing the Igea head and for capturing the face and neck of the bunny (see Fig. 4.9).

Discussion A cylindrical Chebyshev net *requires* diagonal initial conditions and thus the zig-zag tool. In the case of a disk topology, either type of initial conditions may be used, and the choice is one of aesthetics, versatility, and convenience.

The zig-zag approach is otherwise preferred because of its versatility. One example of a recurring strategy with this tool is to pass a diagonal through a high curvature region: Knowing that curvature will decrease in magnitude away from the diagonal, the designer specifies a high shear along the diagonal, thus allowing the mesh to reduce in shear when integrated outward.

With either of the tools, the user selects the resolution of integration, and specifies whether the shear bound should be respected. Integration then continues as far as possible subject to the initial conditions, boundary of the guide form, and (optionally) the shear bound.

Both integration methods have been optimized using spatial hashing for accurate and fast surface intersection tests. This allows interactive exploration of various integration options in order to find a good wire mesh patch that can be used in Phase II of the design process.

Drawing curves These tools require the user to draw curves on the guide surface. In our implementation we allow for three simple approaches: (i) the designer positions a plane, and the algorithm computes the plane-surface intersection; (ii) the designer picks a surface point, a direction, and a distance, and the algorithm integrates out a geodesic; (iii) the designer picks a sequence of points forming a polyline, and the algorithm projects the polyline to the guide form.

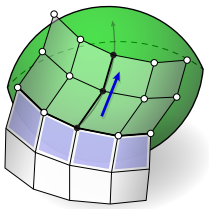
4.4.2 Phase II - Approximating the Guide Form

The second part of the design process allows for modeling tools that *approximate* instead of *interpolate* the guide form. There are multiple reasons why interpolating integration alone falls short: First, interpolating integration typically does not create as much material as desired, e.g., due to exhaustion of initial data. Second, *interpolation* may be too strong a request (§4.3.3). After other editing tools are used, the wire mesh will *approximate* rather than interpolate the guide form.

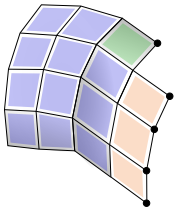
4.4.2.1 Adding Material

New material must be added to a given wire mesh patch when interpolating integration can no longer proceed. The tools described here allow for the creation of new material that respect the Chebyshev conditions at the expense of interpolating the guide form.

Translational surface tool This tool extends an existing wire mesh by propagating parallel to a profile curve drawn on the guide form, but does not additionally seek proximity to the guide form. The designer selects a wire mesh boundary and draws a profile curve on the guide form. The computer then creates additional wire mesh material following the translation surface integration described in §4.3 using the profile guide form and the wire mesh boundary as the two translation curves. This tool is particularly useful for adding wire mesh material along regions of high or oscillatory curvature.



Reflection tool This tool extends an existing wire mesh (blue quads in inset figure) by a small, local addition. After selecting a wire mesh region, the user taps the tool's hot key, and the computer extends the wire mesh by one cell. At corner inclusions, the remaining fourth vertex is uniquely determined by the Chebyshev and planarity conditions (green quad), i.e., just as for the translation surface tool. At boundary edges, the two remaining vertices are uniquely determined by *reflecting* the face across the boundary edge, trivially ensuring compatibility with adjacent extensions (orange quads). If reflection



is repeated without an optimization pass (discussed below), material can be constructed that protrudes far from the guide form.

There are two hotkeys for each of these tools. Both create new Chebyshev material using the specified tool, but one guarantees the shear limit is not violated while the other allows a user specified amount of violation, usually ten percent. Therefore adding new approximate material using these tools either strictly satisfies or nearly satisfies the wire mesh realizability constraints, thereby drastically facilitating the optimizer's task (described below). By allowing the user to create new material which neither satisfies the shear constraint nor interpolates the surface, new material may always be added.

4.4.2.2 Removing Material

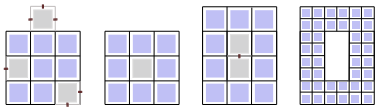


Figure 4.7: From left to right: (1) cutting a boundary cell removes one, two, or three wire edges; (2) cutting an interior cell marks the cell as deleted, but does not remove wires; (3) when two adjacent cells are missing, the separating wire is removed; (4) when the mesh is subdivided, wires are not inserted in deleted cells.

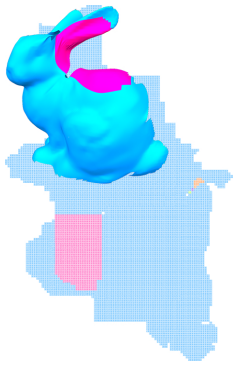
The designer uses the *cutting* tool to eliminate excess material that may otherwise buckle, to trim boundaries, or to punch holes. We do not allow cut edges to be stitched together into a dart as this would introduce valence-three vertices which cannot be fabricated.

The cutting tool is simple to use: the designer selects wire mesh cells, and presses the cutting hot key. The selected cells are then marked as deleted. Wires that bound a live (not deleted) cell are retained, and the remaining wires are discarded.

4.4.2.3 Visualizing and Correcting Parametric Overlap

Repeated cutting and material addition can unintentionally create a design that cannot be cut out of a planar wire mesh, by adding excess material that overlaps in the parametric domain. Throughout the design process, overlapping regions of the parametric domain are brought to the designer's attention via highlighted wire mesh cells, as depicted in the inset figure of the bunny.

Detecting cells that overlap in the parametric domain is a straightforward exercise in reference counting. Because the wire mesh has strictly regular grid connectivity, each cell is easily indexed by integer Cartesian coordinates; when two or more cells have identical coordinates, they overlap.



Our interface automatically detects and highlights, but does not prevent parameteric overlap. We have found the ability to temporarily create overlaps to be an indispensable discretion during the design process. We benefited from freely extending material, temporarily ignoring the highlighted overlaps, and later choosing whether to correct them by cutting from the new material, or from the older overlapping region. The inset shows that the wire mesh on the left ear of the bunny and on its back refer to the same region in the parameter domain, so the artist must make a choice. Similar choices were made in many of our examples (see Fig. 4.9), where the designer traded: the entire left arm of the *armadillo man* for the shell on its back; the bump on the *Stanford bunny's* back for the exterior of its left ear; and the back of the *Igea* head for the front.

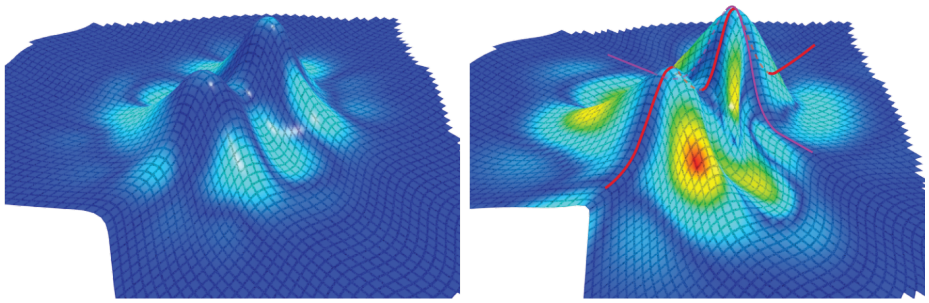


Figure 4.8: Unconstrained global optimization finds a wire mesh close to the target (left), while fixing the wire mesh along two curves (in red) as hard constraints during global optimization produces large deviations from the target (right). Deviation from the target is colored from blue to red.

4.4.2.4 Optimization and Form Shaping

Due to the global nature of Chebyshev nets, our design workflow uses a global optimization approach to improve the shape quality while ensuring the realizability of a design. The optimizer is intended to make a quick calculation that does not impede the interactive, multi-faceted system of tools afforded for design. To use the optimizer, the designer paints weights onto the guide surface to indicate where a close fit should be prioritized. The optimizer seeks to balance the quality of the fit against the fairness of the wire mesh. Global optimization with no hard constraints is essential to find a satisfactory result as illustrated in Figure 4.8.

Solver The constraints are enforced using a generic geometric framework [Bouaziz *et al.*, 2012]. The objective function, together with these constraints, are solved using the augmented Lagrangian technique of [Deng *et al.*, 2013] with one modification: the closeness term is based on distance measured to the guide form, as opposed to displacement from original vertex positions. Briefly, this solver works by introducing auxiliary variables to transform the original problem into separable subproblems with closed-form solutions. This separable structure allows subproblems to be solved in parallel leading to significant speedups on multi-core systems [Bouaziz *et al.*, 2012] and rapid convergence to approximate solutions [Deng *et al.*, 2014].

Problem statement Given a design represented as a Chebyshev net, we write the optimization problem in terms of vertex positions $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$:

$$\begin{aligned} \min_{\mathbf{x}} \;& w_{\text{fair}} F_{\text{fair}}(\mathbf{x}) + F_{\text{close}}(\mathbf{x}) \\ \text{s.t.} \;& \|\mathbf{x}_i - \mathbf{x}_j\| = \ell \;\; \forall (i,j) \in \mathcal{E}, \text{(Chebyshev net)} \\ & |\frac{\pi}{2} - \angle \mathbf{x}_i \mathbf{x}_j \mathbf{x}_k| \leq \gamma_{max} \;\; \forall (i,j,k) \in \mathcal{A}, \text{(Shear limit)} \end{aligned}$$

where the function F_{close} penalizes the deviation between the mesh and guide surface, F_{fair} measures the fairness of the mesh, $\mathcal{E}$ is the index set of vertices that lie on a common edge, $\mathcal{A}$ is the index set of vertices that form a corner of a quad face; ℓ is the constant edge length, γ_{max} is the maximum amount of shear in radians, and w_{fair} is a user-specified positive weight to control the tradeoff between

closeness and fairness. F_{fair} is a quadratic energy defined using the second order difference of vertices

$$F_{\text{fair}}(\mathbf{x}) = \sum_{(i,j,k)\in\mathcal{F}} \|\mathbf{x}_i - 2\mathbf{x}_j + \mathbf{x}_k\|^2,$$

where $\mathcal{F}$ is the index set of three consecutive vertices that lie on a common wire. Such a fairing term inhibits the wire mesh from folding onto itself. F_{close} is the weighted sum of squared distance from the mesh vertex to the guide form

$$F_{\text{close}}(\mathbf{x}) = \sum_{i=1}^{n} W(P(\mathbf{x}_i))\|\mathbf{x}_i - P(\mathbf{x}_i)\|^2,$$

where $P(\mathbf{x}_i)$ is the closest projection of $\mathbf{x}_i$ onto the guide form, and W is a local weight function painted onto the guide form by the user, or a global constant when no local weights are specified.

Closeness term The closest projection $P(\mathbf{x}_i)$ of a vertex of the wire mesh is approximated using a signed distance field. This provides significant speed improvements to the optimization. The field is precomputed on a very high resolution grid (with 20% padding) to capture all the details of the guide form. To emphasize important features of the guide form during global optimization, the local weights $W(P(\mathbf{x}_i))$, are painted onto the guide form vertices and linearly interpolated across the faces.

Discussion The optimization has three parameters which may be changed at any point during the design loop: (i) the global fairness weight w_{fair}, (ii) the surface closeness weights $W(P(\mathbf{x}_i))$, and (iii) the number of iterations to perform. Recall that the design loop tools guarantee that the mesh is Chebyshev and in almost every instance that the shear limit constraint is also satisfied, even though material is being added and removed. Additionally, any new material roughly approximates the guide form. Therefore the modified wire mesh is a good initialization for the global optimization.

Timings We chose a solver which finds approximate solutions quickly. The compute time for a single iteration is 27ms on a modern laptop computer for a mesh with 15,000 vertices. This allows us to achieve interactive performance even for 500-1000 iterations which satisfies the Chebyshev net constraint usually within 1% and the shear limit constraint within 3-5%. The designer therefore has good intuition of the final result at interactive speeds. At the highest subdivided resolution (the edge length is that of the physical wire mesh) we run a final optimization. As for many non-convex optimization problems, there is no guarantee that the solver converges. In practice, however, we observed that 10,000 iterations were sufficient to fabricate the results.

4.4.2.5 Enriching detail via subdivision

To resolve finer details, the designer invokes the subdivision tool. The subdivision scheme globally quadrisects each cell keeping all the original ("even") vertices fixed, while introducing new ("odd") vertices at the cell centroids and edge midpoints. This subdivision automatically preserves the Chebyshev constraints.

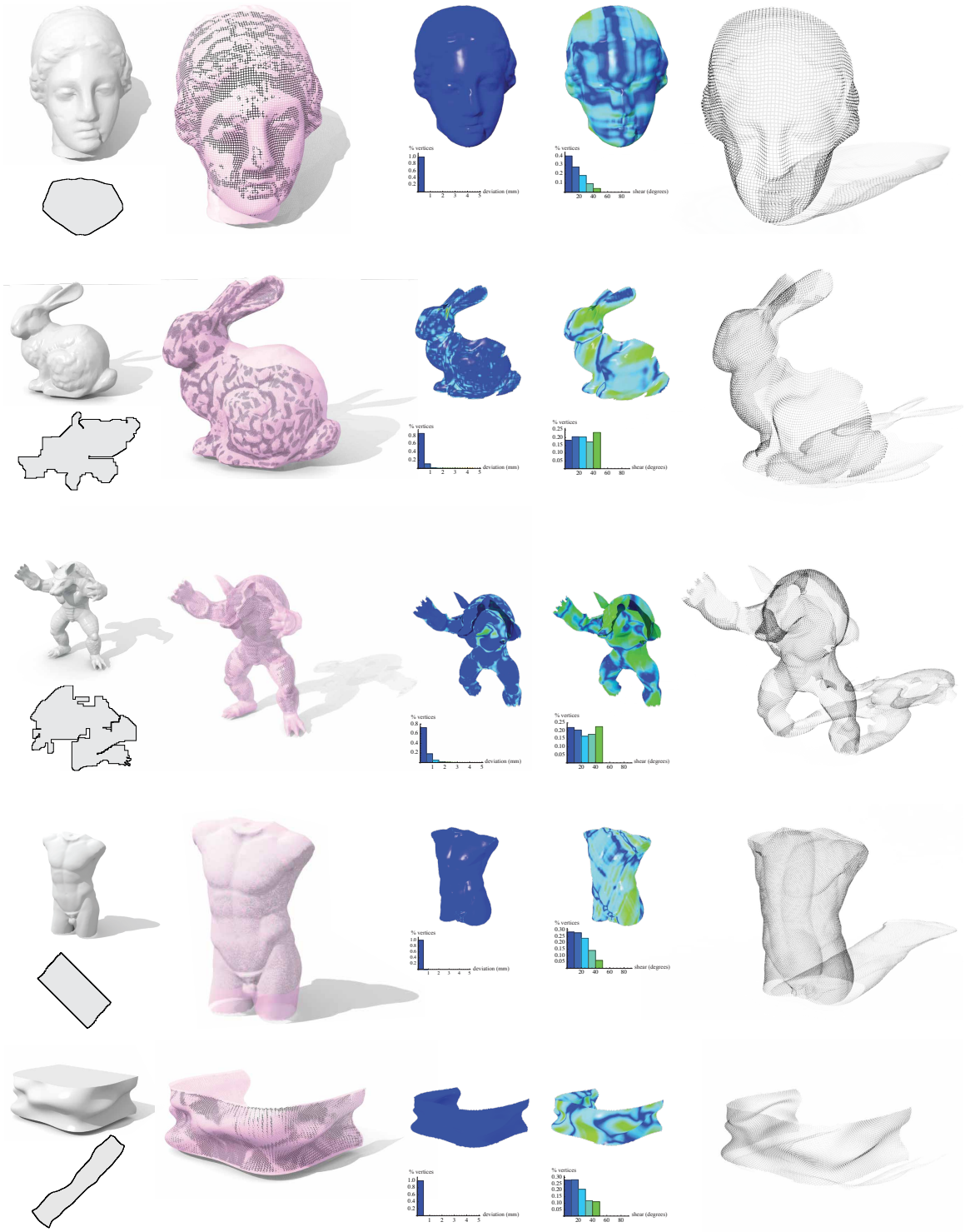


Figure 4.9: Three computer graphics classics, a male torso, and a freeform facade modeled as wire meshes. From left to right: guide surface and final flattened wire mesh, overlay of wire mesh and guide surface, deviation from guide surface, shear distribution, final wire mesh.

Figure 4.10: Wire meshes of a Moai statue (top) and a female torso sculpture (bottom), designed using zig-zag initial conditions to account for their cylindrical topologies.

Figure 4.11: Physical realizations (middle & right) of the female torso (left).

4.5 Results

Our design tool is implemented as a plugin of OPENFLIPPER [Möbius and Kobbelt, 2012]. The accompanying video provides a didactic, visual explanation of the design and interaction process. Figures 4.9 & 4.10 showcase designs created with this process, and Table 4.1 summarizes the associated statistics. These examples demonstrate coverage of large parts of intricate geometries with a single sheet of wire mesh. In all designs, the Chebyshev nets are restricted to a shear limit of $\pi/4$. The optimization automatically distributes shear non-uniformly (see shear distributions, Fig. 4.9, fourth column) so as to simultaneouly satisfy the wire mesh constraints and adhere closely to the guide surface (see deviations plotted in Fig. 4.9, third column).

For the Igea, bunny, and Armadillo models, the domain of each Chebyshev net has been designed interactively using the tools described in §4.4. Through the combination of interaction and optimization, we can capture not only geometrically delicate features such as the bunny ears, but also the global surface structure of the guide surfaces. The Armadillo model is particularly challenging due to the high total curvature resulting from the geometric complexity of its salient features. In the design of this wire mesh, clear tradeoffs have to be made between surface coverage and guide surface adherence. For example, as seen on the inset figure to the left, when trying to retain the bump on the back of the Armadillo the left arm cannot be covered without overlap in the parametric domain.

The male torso, female torso, Moai statue, and facade models have been designed starting from a cylindrical topology. After finding an initial layout of a coarse cylindrical mesh, the design is refined by interleaving mesh edits, subdivision and optimization, to gradually capture prominent features of increasing geometric frequency. While the facade may seem simpler than the torso, there is more curvature variation on the facade model, making the design process

Model	N	K	Time
IGEA	41,700	66.21	10 min
ARMADILLO	66,019	175.96	2 hr
BUNNY	98,239	65.72	2.5 hr
FACADE	66,351	107.04	45 min
TORSO	230,880	124.40	10 min

Table 4.1: Statistics for our design studies. N denotes the number of vertices, K measures the total discrete Gaussian curvature as the sum of all vertex angle defects. *Time* is the total design time including exploration. All numbers refer to the final wire mesh.

more time consuming as the exact amount of material required for a full covering has to be found via interaction. As one bump on the facade is captured better through optimization, material is pulled away from other regions, necessitating addition of material and further optimization. After three levels of subdivision, the optimized meshes fit well to the target surface (see Fig. 4.9, third column). The flat back of the facade model was trimmed away to reveal the final disk topology.

Observe the choice of a diagonal orientation of the wires in many cases and the non axis-aligned domains, thus circumnavigating the restrictions enshrined by the Hazzidakis formula. Indeed, the total discrete Gaussian curvature in all examples significantly exceeds the fundamental 2π limit that is imposed on axis aligned rectangular domains by the Hazzidakis formula. This illustrates that the choice of the right domain is essential when aiming for *single-sheet* coverage of curved surfaces.

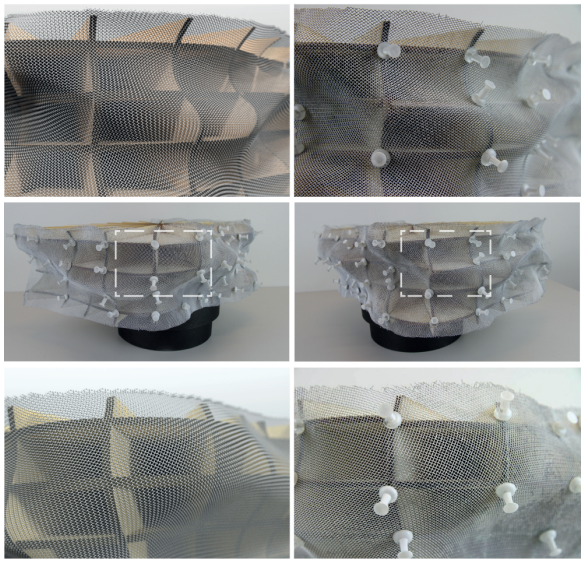


Figure 4.12: The fabricated facade (center), with a comparison between renderings of the designed Chebyshev net (top and bottom left) and photographs of the physical model (top and bottom right).

While numerous artists have manually created compelling wire mesh sculptures of the human body, to our knowledge, no example exists that has cylindrical topology like our torso model, i.e., represents a *complete* section of a body. Previous examples like the one shown in Fig. 4.2 only show the front part with a wire mesh of disk topology.

The facade model illustrates the potential for architectural applications. We cover a complex facade with a single sheet of wire mesh, avoiding patch boundaries with their attendant inconsistencies and visible seems; such contiguous designs improve the visual quality of wire mesh claddings and freeform facades.

Fabrication To validate the agreement of physical wire meshes with digitally designed Chebyshev nets, we fabricated four of the designs. We use 0.34 mm gauge stainless steel wires, woven with a

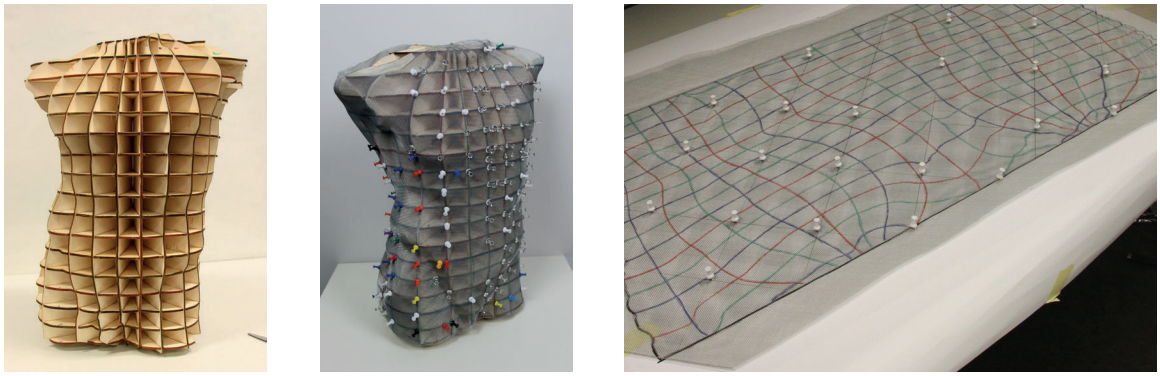


Figure 4.13: Physical fabrication workflow: A 3D scaffold is created by laser cutting intersecting planar pieces (left); the planar wire mesh material is labeled and cut according to the flattened Chebyshev design net (right); the mesh is bend into place according to the labelled curves and pinned to the support (middle); after removing the support, a free standing sculpture of the torso is obtained. The slight tilt of the model results from the non-horizontal lower boundary curve.

plain weave into a wire mesh with 1 mm square openings; correspondingly, our digital designs have at their finest resolution a 1.34 mm centerline spacing. We fabricate in three stages (see Fig. 4.13):

First, we fabricate a scaffold: The wire mesh design is triangulated; to avoid the bias introduced by cutting a quadrilateral by a diagonal, we add a vertex at the center of each quadrilateral. We employ *Autodesk*'s *123D Make* to transform the triangulated mesh into two orthogonal families of planar cross sections, which we laser-cut from 4mm softwood, glue together, and sand at each contour plane intersection (see Fig. 4.13-top-left).

Second, we color the scaffold, referring to an intersection map: We intersect the digital wire mesh against the digital scaffold model; the intersecting faces typically form a network of curves, to which we assign three colors. We then color the contours of the fabricated scaffold according to this color convention. Correspondingly, we digitally map the colored curve network to the parametric plane using the parametric plane construction of §4.4.2.3. The flattened network of colored curves forms our *intersection map*, which we print on paper and transfer onto a large piece of planar shear-free wire mesh (see Fig. 4.13-bottom).

Third, the planar piece of wire mesh is manually bent: We bend the mesh so as to bring the guide curves into alignment with the contours of the scaffold, using push pins to fix the mesh in place. We chose pins with heads large enough to prevent too much slippage from occurring, but small enough to allow the wire mesh to shear. We leave the mesh affixed to the scaffold for 48–72h, allowing time for plastic flow under the applied strain (see Fig. 4.13-top-right), at which time we remove the pins, and the wire mesh, from the scaffold, yielding a freestanding wire mesh (see Figure 4.12).

4.6 Conclusions & Future Work

Computational wire mesh design is a new approach for creating compelling 3D models composed of woven materials. We employ results from the theory of Chebyshev nets to shed light on the intrinsic difficulties of designing with wire meshes. Our analysis calls for a *global approach* with *local control*. We leverage and coordinate the human ability to *understand shape*, and the

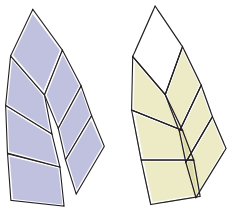


Figure 4.14: Left: before optimization. Right: after opti-

computer's ability to *optimize shape* subject to thousands of constraints.

Limitations A fundamental challenge, both theoretically and practically, of Chebyshev nets is whether a given guide form can be covered in its entirety. Consequently, in our digital design process it is difficult to anticipate the amount of material required to cover the target surface. Our user therefore iteratively adds material, guided by intuition. However, if too much material is supplied at a coarse resolution, subdivision introduces buckles and folds, which are most easily corrected by backtracking the design process. Conversely, if too little material is available near a cut, the optimizer may attempt to close the cut, producing non-local self-intersections; while the fairing energy helps reduce local self-intersections and buckling, it does not prevent more general self-intersections (see Fig. 4.14). Providing more powerful tools to address buckles and self-intersections would accelerate the design process.

Our fabrication process also poses challenges when dealing with the "spring-back" of the physical wire mesh material. A real wire mesh must be "over bent" in certain regions so that it springs back into the desired form. It would be interesting to investigate how to account for this over bending in future work. Even without over bending, producing scaffolds for arbitrary shapes is not easy; *123D Make* constructs strong scaffolds for closed, nearly convex surfaces such as the torso or facade. Generating such scaffolds for highly non-convex shapes, such as the bunny or armadillo, is more difficult. While *123D Make* generates two families of planar contours orthogonal to each other, the works of Cignoni et al. [Cignoni *et al.*, 2014] and Schwartzburg & Pauly [Schwartzburg and Pauly, 2013] construct scaffolds from planar pieces cut at arbitrary angles to one another; these works can hopefully be extended to provide sufficiently strong scaffolds for our fabrication process.

Future work While we make no attempt to directly advance the theory of Chebyshev nets, we hope that by exposing empirical evidence for the rich space of Chebyshev nets that abound under relaxed constraints on surface adherence, our work might inspire new theoretical investigations on approximative nets.

We look forward to extensions of the optimization step that incorporate additional design objectives, such as accounting for the influence of gravity, or optimizing shadows and shade. Indeed, wire meshes are popular in facade applications to reduce solar radiation, where wire thickness, spacing, and shearing all affect shadowing capacity. Extending our software to the design of wire meshes with desirable spatially varying shading capacity could therefore be a powerful tool in architectural form finding.

Chapter 5

Computation Design of Reconfigurables

5.1 Introduction

Using traditional computational tools, the design of a transforming object or a collection of objects transitioning between configurations requires laborious, time-consuming and expensive iterations. This is especially true if relying on physical fabrication merely to determine feasibility. For reconfigurable designs, collisions and interpenetrations pose an editing problem. The solution requires a synergy across space and time lacking in current tools.

We consider the problems unique to the design of objects or collections of objects whose functionality or aesthetic appeal is defined by both the static geometry of participating rigid bodies and their transformations into different configurations, or states. We call such objects *reconfigurables*.

Modeling in isolation one (spatial) component, or one ("temporal") state, of a reconfigurable opens a Pandora box of tempting modifications that invalidate physical realization. For example, exploring appliance and cabinetry choices of a reconfigurable space-saving kitchen, without carefully considering how they affect mutual clearance, could yield an expensive disappointment (see Figure 5.1). Likewise, adding armrests to a transforming bench/table might induce interferences to the transition between states (see Figure 5.2).

We reimagine the typical computer-aided design interface. Rather than optimize for editing static geometries, we treat the transitions of objects between configurations as first class citizens. A designer may interleave edits to an object's geometry in space and edits to its transition through a fictitious graph-based time dimension. All the while, our interactive tools help identify, visualize,

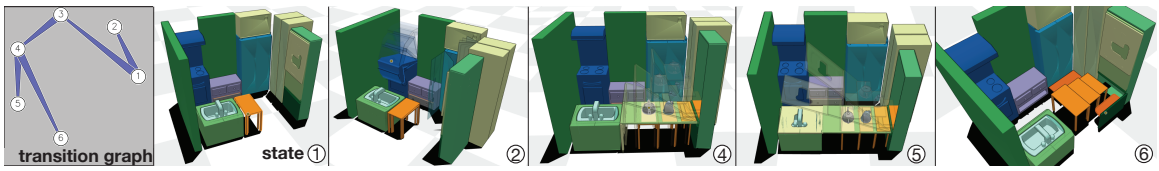


Figure 5.1: A reconfigurable kitchen saves space and maximizes utility. Operating on a graph of seven states (left), our interactive environment enables the designer to add deployable, hidden features, such as a hidden countertop above the range (2), cabinet doors that do not interfere (2), a hidden appliance tabletop (4) with over-sink counterspace (5), and a telescoping eat-in table with swing-out benches (6).

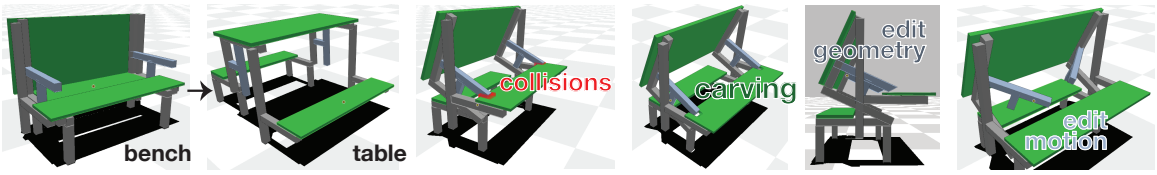


Figure 5.2: Alternative solutions for a reconfigurable that starts as a bench and ends as a table. Solution (a) carves the seat to make room for the arm-rests, (b) reshapes the arm-rests to avoid collisions during transition and (c) arm-rests swing inward during the transition.

monitor and resolve infeasible configurations that may occur along the way.

The first challenge to feasible spacetime editing arises when a seemingly valid edit made in one configuration state creates a new collision in a different state or during a transition. For example, stylizing the handlebars of a bicycle in its unfolded state seems possible, but animating the folding transition reveals new collisions (see Figure 5.4). Relying on manual scrubbing along the timeline and searching via camera controls is far too tedious to work effectively.

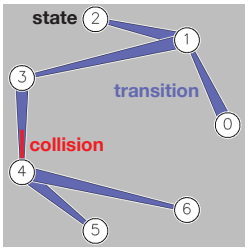


Figure 5.3: Collision alerts appear on the state-transition graph view.

Since collisions might happen at a different time than the designer's current working time selection, the most pressing information is *when* the problem is occurring. Only then is the spatial extent of interpenetrations useful. This order of precedence guides the design of our collision notifications and spacetime collision detection.

The timespans of collision events are identified interactively as the designer makes edits, visualized on each transition's timeline corresponding to edges of a state-transition graph view (see inset). We traverse a four-dimensional bounding volume hierarchy biased toward honing the temporal extent so that notifications immediately appear first on the timeline view. The designer may then jump to this time or preview the collision in a picture-in-picture window, at which point collisions are localized spatially and highlighted in the 3D view.

Some collisions inevitably occur outside the designer's current field of view. The automatic view selection tool transports the designer to an optimal view of the interference. Stroboscopic "ghosts" summarize the reconfigurable's motion near the contact time.

As the designer resolves the collision by editing an object's geometry or configuration, collision notifications interactively update. The designer can track collisions at one moment in time via the picture-in-picture, while making edits at another moment.

Resolving nearly-imperceptible collisions manually can be tedious and ungratifying. If a designer is not sure how to resolve a collision, we provide on-screen hints of edits that would encourage feasibility. Often a designer must make many small or obvious edits only to find that they create new problems elsewhere in spacetime.

Going further than a hint, we can also assist by optimizing over some of the reconfigurable's spatial or temporal degrees of freedom, resolving all slight collisions across spacetime. Our optimization builds on a novel formulation of contact normals, which we derive by extending to 4D a recent contour minimization approach to cloth untangling.

As an alternative to optimizing existing degrees of freedom, the user may also decide that the transformation of one object takes precedence over another object's shape. We propose a novel tool,

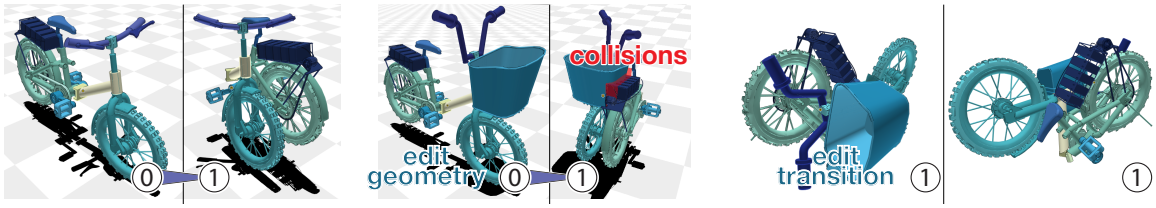


Figure 5.4: The original folding bike is collision free, but adding new handlebars and a basket invalidate this type of folding. The designer changes to a swiveling hinge and finds a collision-free transition.

which combines swept-volume computation with constructive solid geometry to carve away the volume of one object swept along its transformation from the rigid geometry of another.

Through a variety of designs, we demonstrate how the novel

- graph-based time dimension,
- fazy time-then-space collision detection,
- automated picture-in-picture view,
- collision-aware camera optimization,
- collision normal based on spacetime untangling,
- collision resolution hints,
- collision resolution in spacetime, and
- swept volume carving

collaborate harmoniously to afford a fluid interactive design environment. Rapid combinations of our visualization and resolution aids enable fast and creative editing of reconfigurables not possible with traditional tools.

5.2 Related Work

Though computer-aided design (CAD) and computer animation packages are mature and powerful, each on its own falls short when trying to design a reconfigurable.

Optimized for creating intricate static objects, traditional CAD packages offer only simple motion previewing, e.g., a screw twisting into a bolt or a spinning collection of gears. Some commercial tools incorporate interference detection, simply alerting the user if a prescribed motion has created an overlap. For example, a user of SOLIDWORKS can click a button to detect collisions and is alerted via an audible bell sound effect. The detection is not interactive as edits are made, nor does it cluster related collisions. The software does not offer auto-correct or polishing tools. The transition and reconfiguration of an object is not given importance on the same level as the object's surface quality or material strength.

In a complementary manner, computer animation packages *do* place a primary emphasis on the time-synchronized motion of characters and their environments, but time is linear and unidirectional. Our transitions between states are in general a combinatorial graph connecting many states with distinct, potentially bidirectional stretches of time (see Figure 5.1). Animation, by intent, worries about physical accuracy and feasibility of configurations only in so far as they affect perception. For example, MAYA will detect and respond to collisions during a rigid-/elastic-body simulation, but not during geometry modeling or keyframe animating. Indeed, self-interference of an unfolding bicycle or transformer robot may be perfectly acceptable for a film or video game if unnoticeable.

While we do marry concepts from both CAD and computer animation, our primary investigation asks what *new* tools are needed in the context of computational design of *reconfigurables*.

Static shape design The computational design of *static* shapes also benefits from concepts developed for computer animation or computer graphics. A variety of recent works help users find a physically realizable static output shape that achieves a certain functionality (e.g., [Bharaj *et al.*, 2015]) or aesthetic appeal (e.g., [Igarashi *et al.*, 2012; Garg *et al.*, 2014; Schüller *et al.*, 2014]). The focus of this class of works is on providing an interactive tool for finding a somehow "optimal" yet feasible static shape in a single configuration. The quality metrics and feasibility considerations vary depending on the specific application. Some may employ static physics to tighten the design loop, e.g., for designing spinning, balancing toys [Bächer *et al.*, 2014] or structural-sound furniture [Umetani *et al.*, 2012b].

In other scenarios, a shape's quality is determined also by its dynamic behavior, e.g., designing optimal paper airplanes requires consideration of aerodynamics [Umetani *et al.*, 2014]. The interactive garment design tool of Umetani et al. [2011] previews draping dynamics, detecting and resolving collisions as garments contact a mannequin. Unlike our proposal, these tools either ignore inter-object collisions or resolve them in a weak sense à la computer animation: e.g., collisions are important to garment design in so far as they help capture or predict the draping behavior, but collisions are not used explicitly to determine feasibility of a design.

Other works consider a static arrangement of static shapes according to their aesthetic or functional appeal without worrying about the feasibility of possible transitions between arrangements. For example, the furniture in the arrangements of [Merrell *et al.*, 2011b; Yu *et al.*, 2011; Liu *et al.*, 2015] are expected to remain in their assigned place, unlike our reconfigurable apartment design in Figure 5.19.

Fabricating animations While our work is not the first to consider physical constraints during computational design, previous works primarily focus on a very specific class of animated objects. Some methods simply ignore collision during feasibility analysis: e.g., when designing toys [Bächer *et al.*, 2012; Ceylan *et al.*, 2013; Skouras *et al.*, 2013] or mechanical linkage assemblies [Thomaszewski *et al.*, 2014; Bächer *et al.*, 2015]. Similar to the use of collision resolution for static garment design [Umetani *et al.*, 2011], Skouras et al. weakly resolve collisions to design balloons that inflate from a flat configuration [2012b; 2013]

Existing works in the realm of reconfigurables consider and take advantage of a heavily constrained space. For example, Li et al. deform shapes to stack vertically on other instances of the same shape [2012], and Zhou et al. decompose and transform arbitrary objects into rectilinear boxes [2014]. Interlocking and twisting puzzles are an interesting subclass of reconfigurables [Xin *et al.*, 2011; Sun and Zheng, 2015]. For both types of puzzles, managing collisions during design is important, but also simpler due to the limited design space.

We are inspired by the prototyping tool for hinge-based reconfigurables [Koo *et al.*, 2014]. By working only with simple box-based shapes, Koo et al. can dramatically simplify collision handling and contact relationships. We expand this scope by examining computational design of reconfigurables composed of arbitrary rigid parts.

Interactive motion editing At a technological level, our spacetime collision detection and resolution shares components and motivation with the fundamental path-planning problem in robotics (see e.g., [Kavraki *et al.*, 1996]). However, we are not interested in the fully automatic computation of the path of objects to achieve a certain high-level goal, but rather to provide an interactive tool for editing objects and their transitions between configurations simultaneously.

In computer animation, the spacetime constraints paradigm attempts to reduce animator effort for creating realistic looking animations of a character hitting certain configurations at certain moments in time [Witkin and Kass, 1988]. Advanced spacetime methods incorporate contacts and collisions [Popović *et al.*, 2000], but the goal is inevitably to achieve a desirable animation arc. Recently, interactive motion editing also finds interesting applications in the physical world, e.g., designing paths for drone cinematography [Joubert *et al.*, 2015]. While this interface incorporated physical flight constraints, possible collisions are ignored.

During reconfigurable design, our notion of time is artificial. Helping the user find a clear transition path is important, but there is no required concept of momentum, velocity or forces.

Modeling with collisions While we do not intend to model true dynamics, our fictitious time dimension is *not* to be confused with the use of interaction time to exploit collisions for modeling. For example, Harmon et al. essentially treat each user edit during free-form modeling as the next time step of a simulation [2011]. Modeled objects respond to new collisions according to (possibly non-physical) energy-minimizing forces. Similarly, Bernstein and Wojtan exploit the hysteresis between each edit to conduct constructive-solid geometry operations on artifact-ridden geometry [2013]. In contrast, our collision resolution operates directly on a 4D object–the spacetime trajectory—while these earlier works operate on a 3D, spatial object. Their works incorporate (quasi-)time by sequencing spatial collision resolutions. By operating directly on a 4D object, and avoiding the "orientation" of time in a sequential process, our response maintains temporal symmetry. We show that computing a resolution normal for a spacetime trajectory can be achieved exactly, robustly, and simply by integrating over time the contour minimization normals proposed in earlier work for the purely spatial setting.

Rather than liken our work to those that treat modeling as a time-integrated simulation, we interpret our problem as high-dimension analog of "untangling cloth" [Baraff *et al.*, 2003; Volino and Magnenat-Thalmann, 2006a]. These works attempt to remove intersections of surfaces in $\mathbb{R}^3$ (presumably resulting from a cloth simulation) *without* knowledge of velocities or previous configurations. One can interpret a standard surface modeling tool, such as MAYA, as a tool suitable for the interactive, albeit manual, untangling of surfaces. Under this philosophical analogy, our work may be viewed as a conceptualization of how modeling packages might be extended to treat spacetime hypersurfaces in $\mathbb{R}^4$. Going beyond manual interactive tools and aids, we also propose automatic spacetime untangling.

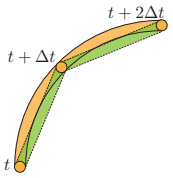
To achieve this we construct a four-dimensional *spacetime* bounding volume hierarchy for intersection detection. In contrast, there are many previous works for detecting collisions instantaneously [Allard *et al.*, 2010] or in *continuous time* over short localized spans of time [Tang *et al.*, 2011; Wang *et al.*, 2012].

5.3 Kinematics and Collision Intervals

The reconfigurable consists of a collection of objects (or parts) that *transition* between various *states*. Abstracted as a graph, states and transitions are nodes and edges, respectively. A visualization of this graph provides a quick overview of the entire reconfigurable (see Figure 5.3). The graph view quickly indicates to the designer if new collisions appear and allows the designer to monitor collisions within one transition while editing another.

Because we are not interested in dynamic effects such as inertia, we parameterize the motion of an object during a transition along a fictitious time interval $[0, 1]$. Each object is assumed to be a solid bounded by a triangle mesh. In our implementation, objects are limited to rigid motions, i.e., all vertices of one object follow the same translations and rotations during a transition.

Borrowing from early work in collision detection [Cameron, 1990], each solid in motion can be viewed instead as a static object in spacetime, occupying a spacetime volume (STV) in $R^3 \times [0, 1]$.



To simplify collision detection, we discretize each vertex path piecewise linearly between evenly sampled time intervals Dt. We may choose Dt sufficiently small to control the tolerance between the true continuous path (orange in inset) and the discretized path (green). The resulting piecewise-linear trajectory serves as the *spacetime proxy* for collision detection.

Our collision detector accepts as input spacetime proxies for all objects of a transition. It identifies pairs of triangles that interfere in spacetime. Each such pair is recorded by its spacetime axis aligned bounding box [Cameron, 1990], and a reference to the two involved objects. We call such a record a *collision interval* (CI). In general, multiple CIs output from the detector may overlap each other in spacetime. We simplify further by merging overlapping CIs, until the CIs are sufficiently coarse (see §5.4.1). A merged CI may now reference multiple involved objects.

CIs lay the foundation for all visualization and resolution tools that our design framework provides. The following sections will make use of CIs extensively in order to support the development of tools necessary to visualize collisions in both space and time as well as help the user to resolve those collisions.

5.4 User Tools

Our investigation is driven by the aspiration to facilitate an interactive, nonlinear, free flowing design process. As the designer alters an object's geometry, state configuration or transition motion, the design environment provides interactive feedback about possible invalid states or transitions. The design environment also guides the designer to resolve these errors by an array of tools that range from informative and unintrusive, yielding control of alterations to the designer, to automatic intervention, alleviating tedium. Our research goal is to understand both the interaction metaphors as well as the under-the-hood infrastructure best suited for interactive, assistive design. Discarding potential tools is just as important as retaining the most effective ones. We present the set front-end tools that we have found to be most effective and generalizable to a broad class of reconfigurable design problems.

As depicted in Figure 5.5 and the accompanying movie, our design tool builds on keyframe animation to sculpt motions (translations and rotations) of objects, using a timeline and 3D viewer metaphor. This keyframe timeline corresponds to a single edge/transition in the graph view seen in Figure 5.3. Object transitions within the timeline are represented by keyframed translations along a cubic Beziér spline and spherically interpolated keyframed rotations.

A typical design session begins with an arbitrary arrangement of objects. Additional objects can be added and removed from the scene dynamically. In our research implementation, we externalize those static-shape modeling tasks that are well established and easily accessible in commercial tools, e.g., MAYA's modeler. In the following, we describe the tools that make up the design environment roughly ordered from least to most interventionistic.

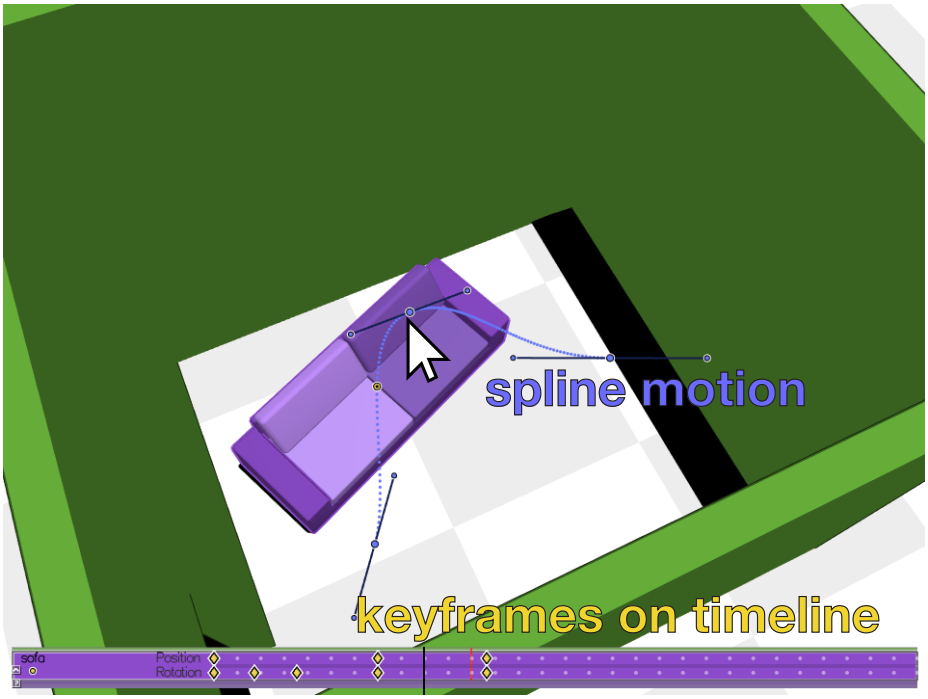


Figure 5.5: Using our modeling interface, a designer choreographs the motion of a couch moving inside a reconfigurable apartment (green walls).

5.4.1 Visualizing Collision Intervals

Complicated reconfigurables are difficult to navigate virtually. As objects transition between states, their relative spatial arrangements are functions of time. Therefore detecting overlaps and collisions between objects *manually*, i.e., via visual inspection, is a tedious process of view manipulations and timeline scrubbing.

We automate the process of identifying and isolating collision intervals (CIs). This process runs continuously in a background thread while the designer makes edits (§5.5). To direct attention to problem areas, the design environment's 3D viewer displays optionally-pulsating transparent boxes over the spatial extent of each CI (see Figures 5.6, 5.8) and the timeline highlights the corresponding temporal extent (see Figures 5.6, 5.7). The 3D viewer highlights only those CIs that overlap the selected moment in time, corresponding to the time marker on the timeline.

If a CI is not easy to interpret from the current viewpoint, a hotkey provides invocation to view optimization (§5.4.3) for rapid optimal focus on a CI of interest. The corresponding spatial and temporal highlights, combined with view optimization, enable the designer to quickly navigate, reason, and act on problems that arise during design revision.

We found that displaying the raw CIs produced by the collision detection routine produces a picture that is in general too busy for a designer to parse. We therefore consolidate CIs that overlap in spacetime: CIs with overlapping or adjacent temporal extents, and spatial extents, merge into one (conservative) CI.

Although considerable consolidation is useful for visualizing temporal extents on the timeline, the same degree of consolidation produces spatial extents that are too conservative (large) to provide

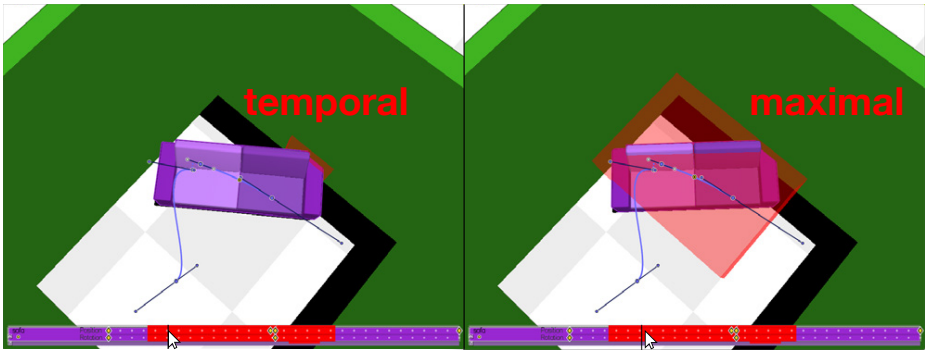


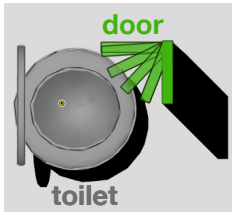
Figure 5.6: Left: consolidation of spatial extents over small stretches in time give us more context over collisions compared to, Right: maximal consolidation in space. Both: The timeline always enjoys maximal consolidation of CIs.

any useful context over colliding objects. Thus we consolidate to different extents for the timeline and 3D viewer. While the former enjoys maximal consolidation, the latter is limited to consolidations that keep the temporal extent no longer than a small stretch of time. This ensures that all visualized pulsating highlights encompass colliding regions that occur at the frequency of a single visual frame (see Figure 5.6).

5.4.2 Ghosting

Inspired by Snibbe [1995], we optionally display "ghosts" of any transitioning object. These allow the designer to see the object's transition without actively scrubbing the timeline. Formally, ghosting is computed by projecting isotemporal slices of object's spacetime volume onto the current 3D view at the selected time. We render the object at regular samples over time, with progressively decreased opacity away from the selected time [Everitt, 2001].

Ghosting is particularly useful for laying out objects in tight-fitting spaces, allowing the designer to conservatively reason about potential collision states without repeatedly scrubbing the timeline (see Figure 5.7). In the inset figure, the designer reasons about the layout and clearance of a swinging door and toilet obstacle.



5.4.3 View Selection

Visual occlusions due to perspective projection may hide collisions in the scene. Other collisions may simply not be in the current field of view. To help the user inspect problem areas, we propose a method for selecting optimal camera views of collisions. Our interactive design loop requires a fast optimization that considers the dynamic objects. Though analogous in spirit to previous works on optimal view selection [Secord *et al.*, 2011], we have neither the luxury of precomputation nor the convenience of a single, static shape.

We assume a rigid camera with three translational degrees freedom and three rotational degrees of freedom. Assuming a "fixed up axis" (e.g., as in AUTOCAD, MAYA, etc.), we may omit the rotational degree of freedom *twisting* about the viewing axis.

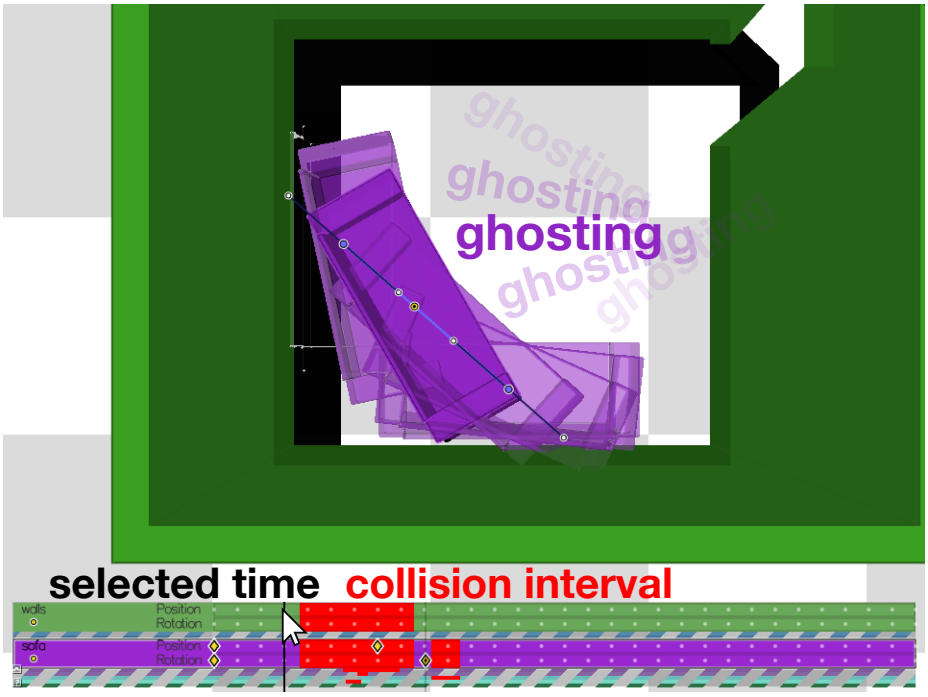


Figure 5.7: The designer makes use of "ghosting" to relocate the couch around a tight corner.

Given a collision interval selected via the timeline, we first identify the center of the collision's spatial extent. While one could imagine various weightings on the spatial contact points to compute a center position, we observe that simply taking the barycenter of the spatial bounding box $\mathcal{X}$ containing all contact points found within the collision interval is consistent, predictable and robust. The camera is constrained to place this center position along its viewing ray: two rotational parameters and the distance from the center remain unknown.

We select the smallest distance containing the bounding box $\mathcal{X}$ completely in view for any angle.

Finally, to optimize the rotational degrees of freedom, we consider two scenarios: 1) collisions involving two objects and 2) collisions involving more than two objects.

Two objects Brief collisions involving two rigid objects $\mathcal{A}$ and $\mathcal{B}$ are well approximated by their relative translation. Given the *rigid* motions of $\mathcal{A}$, we define the direction of approximation motion during the collision interval with temporal extent $[t_0, t_1]$ to be simply the difference in center of mass at the end and beginning of the interval: $\mathbf{r}_\mathcal{A} = \mathbf{A}(t_1) - \mathbf{A}(t_0)$, and equivalently for $\mathbf{r}_\mathcal{B}$.

A very degenerate (non-optimal) view would be along $\mathbf{r}_\mathcal{A}$ or $\mathbf{r}_\mathcal{B}$: in this view one object occludes its own motion and likely also the collision with the other object. The ideal view places the approximate motion of both objects in the view plane: the view direction should be perpendicular to both $\mathbf{r}_\mathcal{A}$ and $\mathbf{r}_\mathcal{B}$. In other words, the viewing direction should be parallel to the cross product of these two directions: $\pm(\mathbf{r}_\mathcal{A} \times \mathbf{r}_\mathcal{B})$.

In this case, the optimization boils down to choosing between two views (see Figure 5.8). This decision is made based on a measure of the collision's occlusion. The exact occlusion induced by these and other objects meshes is difficult and expensive to measure. Our view selection is only useful if instantaneous. Therefore, we approximate occlusion via the traditional real-time rendering

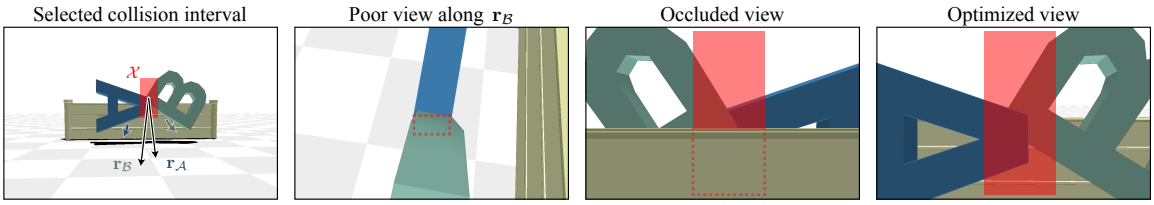


Figure 5.8: After selecting a collision interval on the timeline, a user can jump to an optimized view by hitting a hot key. Views along the trajectory of an object (e.g., $\mathbf{r}_\mathcal{B}$) are poor. Views orthogonal to the movement of both objects are ideal. Of these two views, we choose the one with the least occlusion.

pipeline.

We render the scene with an opaque bounding box $\mathcal{X}$ around the collision's spatial extent and count the number of visible pixels of the bounding box. We implement this by rendering the scene once to set the depth buffer and then rendering only the bounding box again while counting pixels using the hardware "occlusion queries" (`GL_SAMPLES_PASSED` in OpenGL). In our experimental setup, this outperformed rendering with depth culling and counting pixels (even when using a GPGPU parallel reduction).

Many objects Now, let us consider the multi-object collision scenario. In this case, the approximate directions of all objects will not define a unique ideal viewing axis: any view axis will be non-perpendicular to some direction. We experimented with deriving a least-squares optimization to find an "as-perpendicular-as-possible" viewing direction, but this frequently resulted in degenerate views. Instead, we propose that in multi-object scenarios the prevention of occlusions outweighs the importance of finding a perfect direction relative to the object motions.

Optimizing over all views is infeasible, so we opt for a Monte-Carlo approach: we sample many (>1000) random views and choose the best according to the number of visible pixels from the collision box as described above. For scenes with an obvious "ground" (e.g., the space-saving kitchen in Figure 5.1), we restrict the sampling to the northern hemisphere. If the camera control is a trackball (i.e., without a fixed up axis), we uniformly sample random orbiting rotations as quaternions [Shoemake, 1992]. If using "fixed up" camera controls, we uniformly sample directions via points on the unit (hemi-)sphere [Shao and Badler, 1996].

5.4.4 Picture in Picture

Side effects are a common occurrence when trying to resolve collisions of a reconfigurable. While the designer focuses on repairing a particular CI, a new collision may form elsewhere in spacetime. Recognizing this, then scrubbing back and forth between collisions and their side effects hinders productivity.

Our tool activates a picture-and-picture (PIP) view when a side effect collision occurs. As seen in Figure 5.9, the PIP is view-optimized (see §5.4.3) on the side effect CI. The corresponding temporal position of the PIP is marked by a secondary yellow marker on the timeline. As the designer makes edits in the design the PIP view updates accordingly. When the designer clicks on the PIP window, the main and PIP views are swapped, accelerating the typical back-and-forth workflow required to achieve multiple conflicting spacetime corrections.

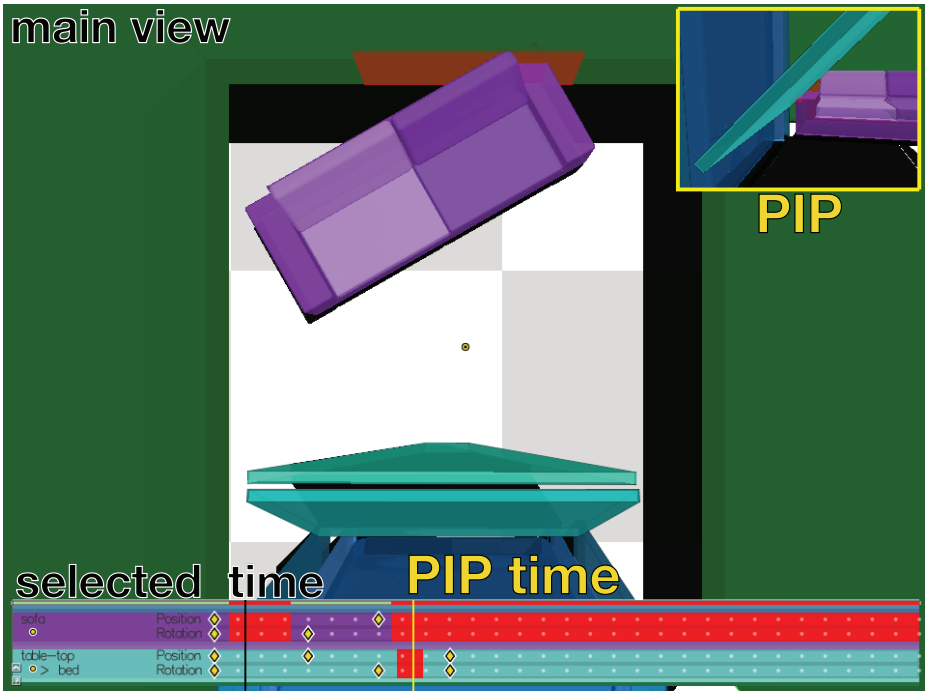


Figure 5.9: As the designer attempts to resolve the current collision of the sofa against the wall, another collision is triggered and shown in the PIP view.

Triggering PIP Modification of the reconfigurable triggers background collision detection (see §5.5), which yields a set $\{I_1, I_2, \ldots\}$ of spacetime CIs, $I_i \in \mathbb{R}^3 \times [0,1]$. By comparing the sets before and after the modification, $\{\hat{I}_i\}$ and $\{I_i\}$, respectively, we determine which CIs (i) remain unchanged, (ii) disappeared, (iii) appeared, or (iv) changed spatiotemporal extent. Whereas cases (i) and (ii) do not call for the designer's attention, cases (iii) and (iv) correspond to side effects that the designer must review.

We first considered displaying multiple PIPs for the multiple CIs, but this clutters the user interface. Instead, we prioritize CIs and alerting the designer to the most critical side effect via a single PIP. This decision follows and reinforces the typical design workflow of attending to the most critical problems first.

We first prioritize *new* CIs, case (iii), over *altered* CIs, case (iv). Among new CIs, we prioritize *longer* time extents over *shorter* durations Among altered CIs, we prioritize *extending* over *contracting* durations. The sorting of PIPs based on duration is intended to match the intended workflow of honing in on CI time and then space.

The PIP, like all 3D views, is a slice through spacetime at a so-far-undetermined fixed time. Keeping with our design philosophy of attending to the most critical sections first, the time instance chosen corresponds to *when* the collision is *most severe* within the CI. Classically, one would use a measure of penetration depth to measure the severity of a collision region. However, measuring penetration depth usually requires the use of distance field methods, which introduce substantial computational overhead and are not suitable for interactive applications when geometry deforms significantly [Teschner *et al.*, 2004]. Thus, we rely on our notion of intersection energy (5.1), detailed in §5.6.2, as a measure of severity. Having selected a most severe time during the CI, the PIP view

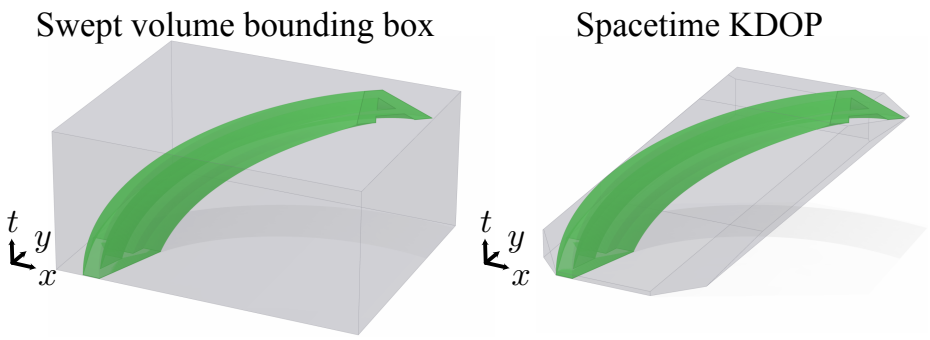


Figure 5.10: The 2D object A rotates over time. The bounding box of the swept volume extruded into time is very conservative (left). A k-DOP in spacetime hugs the spacetime volume tighter.

shows the spatial collision configuration using the view selection algorithm (see §5.4.3).

Finally, the design can trigger the PIP view manually via a hot-key while hovering over a CI on the timeline. This enables easy navigation between CIs (see accompanying video).

5.5 Collision Detection

As the user edits and revises the design, our tool continuously runs a spacetime collision detection thread. This spacetime collision detection thread enables the user tools in the previous section to bring attention to any design trajectories that may interfere with one another, highlighting first *when* in time collisions occur before computing *exactly* where they occur.

Given a reconfigurable, our goal is to identify collision intervals (CIs) in spacetime where pairs of objects overlap. The following basic algorithm identifies all overlaps between piecewise-linear spacetime proxies. In order to achieve interactive rates for continuously modified designs, we use a spacetime k-DOP bounding volume hierarchy (BVH). For complex designs, recomputing the hierarchy on every edit would prohibit interactivity. We exploit the locality of design edits by *updating* and *traversing* only modified portions of the BVH.

5.5.1 Spacetime k-DOP

A wealth of bounding volume (BV) types have been explored [Klosowski *et al.*, 1998]; we adopt k-DOPs [Klosowski *et al.*, 1998], whose popularity stems from their easy construction, efficient update and comparison operations. A k-DOP bounds a volume by taking the intersection of tight fitting half-spaces associated with a fixed family of k axes. The question arises, how large should k be? In three dimensions, popular choices are 6 (an axis-aligned bounding box) and 26 (choosing the corners, edge midpoints, and face barycenters of the unit cube to form axes). If we interpret $\mathbb{R}^3$ instead as two spatial dimensions plus time (e.g. $\mathbb{R}^2 \times [0,1]$), then we may understand the effect of this choice in spacetime (see Figure 5.10).

For our spacetime k-DOP, we consider a variety of generalizations. We could take the axis-aligned bounding box in four-dimensional spacetime: 8-DOP. On the opposite end of the spectrum, we could take an 80-DOP with halfspaces corresponding to all combinations of spatial and temporal axes. A good balance is to augment the generous spatial 26-DOP halfspaces with two temporal

halfspaces, resulting in a 28-DOP.

We implemented our bounding volume hierarchy generically to these choices. Our experiments reveal that the 28-DOP slightly outperforms the 80-DOP, and both greatly out perform the 8-DOP. Although the 80-DOP provides fewer false-positives, the spacetime proxies themselves only span small segments of time where false positives are rare. Therefore, we avoid extra computational cost of testing so many plane equations, using only the 28-DOP (on all our examples).

5.5.2 Constructing the Bounding Volume Hierarchy

We construct binary tree BVH for each object once at initialization. We employ a top-down tree construction strategy that recursively splits our BVH until each leaf contains a single spacetime proxy (a spacetime triangle). At each non-leaf node, the splitting procedure creates a splitting plane all the 4D vertices of the spacetime proxies. This plane is chosen orthogonal to the axis of the coordinate with maximum extent, centered at the median value. Note that this splitting does not necessarily create two disjoint children, i.e., it may result in children whose BVs overlap if a proxy is assigned partially to each halfspace. We experimented with increasing the branching factor to the tree: from binary to ternary and octonary, but we did not experience any performance gains.

5.5.3 Local, Lazy Refitting

As the designer edits the reconfigurable, the structure of each object's BVH remains intact, while the extents of the affected k-DOPs are updated. Since the designer typical makes structured, localized and continuous edits, we found that refitting is usually much faster than rebuilding the tree from scratch.

The animation keyframe user interface inherently leads to a temporal locality of revisions. Further, the use of a (mouse-)pointer-based interface typically results in spatially-localized editing. In general, only a small subset of the entire spacetime reconfigurable is altered with each designer gesture. Marking the affected spacetime proxies as "dirty," we exploit the locality of editing operations by refitting and "cleaning up" only those dirty k-DOPs impacted by dirty proxies. This local refitting allows for interactive collision detection, even for many objects in large scenes.

Correspondingly, pairwise BVH traversal need not be carried out over the complete set of all BVHs. Rather, in the spirit of *kinetic data structures* [Guibas, 1998], only comparisons involving the "dirtied" subtrees of a BVH need to be checked against the clean subtrees.

5.5.4 Visualizing Collision Intervals

In order to provide the designer with the most relevant information immediately, we give precedence to first visualizing *when* collisions occur, so that the design process may benefit from immediate high-level feedback when edits invalidate trajectories of objects in the scene.

When Collisions Occur Although BVH traversal schemes typically proceed top-down based on a stack, we opt for a breadth-first traversal based on a queue. Each level of the BVH that is traversed in this way provides an initial "preview" to where potential collisions lurk. These previews are displayed on the timeline, with progressively increasing opacity at deeper tree levels (see Figure 5.11). While the local, lazy refitting typically allows for nearly instantaneous completion of all collision

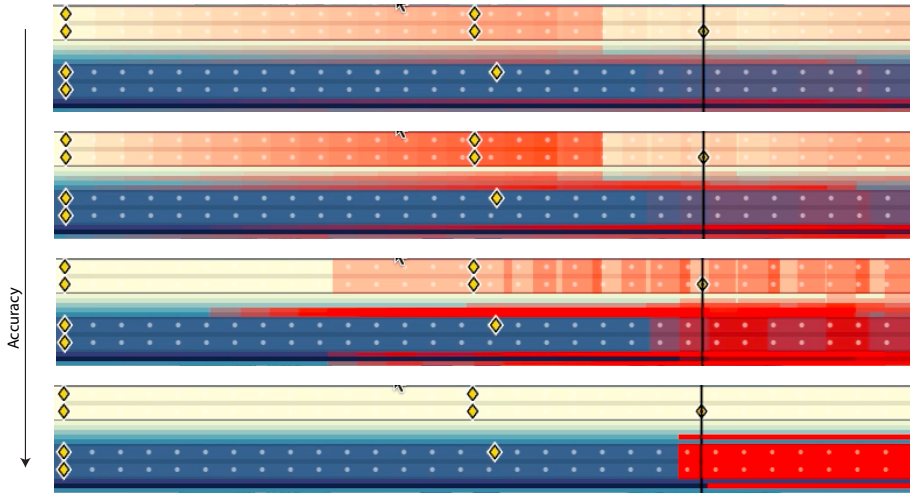


Figure 5.11: Accuracy of the visualized CIs increase as deeper nodes in the BVH are processed. Opaque CIs are the most accurate.

detection, the breadth first traversal with previews further reinforces the impression of immediacy even for operations that less localized, such as the automatic resolution described in §5.6.6.

Where Collisions Occur After honing in on the time extent of the CI, the designer can jump via the timeline or picture-in-picture to a moment within the CI. The BVH hones the spatial extents and returns a set of candidate spacetime primitives, involving two triangles a and b over a time step Δt. The final CIs are built by conducting continuous collision detection (CCD) over the time interval Dt. The solution to the CCD procedure provides a $dt \subset Dt$, where dt is the time-extents when a and b are overlapping. The CI stores this dt as well as the spatial extents of the collision region between a and b within dt. We store mappings between CIs and the objects to which the primitive a and b belong, allowing for fast queries of CI and their objects.

5.6 Spacetime Collision Resolution

Our automatic resolution of collisions is set apart from earlier work on collision response by its focus on altering spacetime configurations, rather than the more common goal of computing forces or displacements in the context of forward time integration.

To develop an automatic collision resolution method we must first define a notion of a four-dimensional constraint gradient or *contact normal*. For a spacetime point on our spacetime shapes this is the direction of the admissible region of configuration space: no interpenetrations.

5.6.1 Less Decent Descent Directions

Before embarking on our normal derivation, we summarize our early failures with various alternatives. An obvious approach is to move objects according to the spatial normal evaluated at the first instant of contact. Similar to our chain rule differentiation with respect to the UI, this direction can be projected onto the UI degrees of freedom. Beyond the philosophical question of what is a “first”

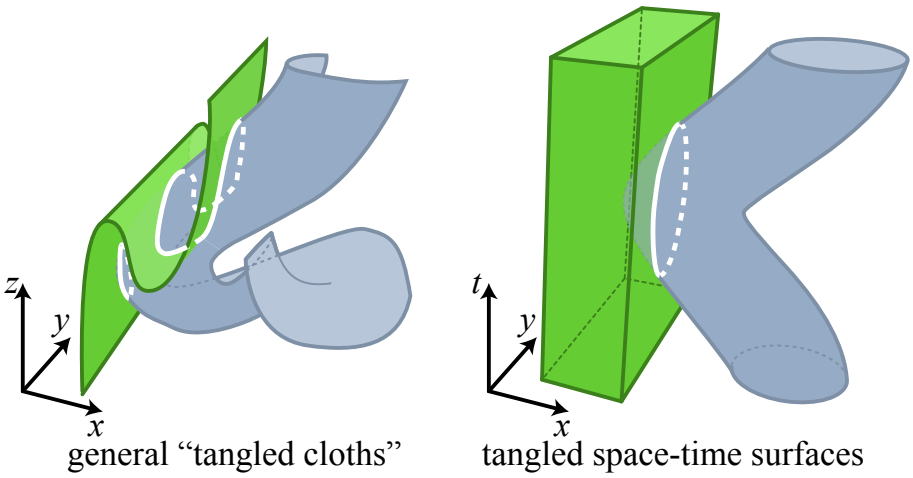


Figure 5.12: Left: intersection of general 2D surfaces in 3D. Right: intersecting spacetime volumes of 2D curves.

instant of contact in a fictitious time dimension (and why is “first” better than “last,” see Figure 5.14), we found this normal to be very sensitive, or “noisy,” with respect to small perturbations to position or shape.

To restore temporal symmetry we considered averaging the contact normals at the first and last instants contact, but noise remained. To reduce the noise, we considered averaging over all contact normals throughout the collision duration, but the result was not useful: contact normals at a pair of overlapping mesh primitives, belonging to two already overlapping objects, can point in essentially any direction, and do not provide a useful direction for resolving the intersection.

Another alternative we considered was not a definition of the contact normal per se, but a different approach to collision response altogether. We considered treating the collision interval as a physical simulation, *plowing* through the collisions by integrating forward in time. To do this, we experimented with a traditional linear complementary problem (LCP) formulation of collision response, but found that the position-based variant did not always have a feasible solution [Erleben, 2004] while the velocity-based variant was not guaranteed to resolve all collisions nor stay close to the designer’s intended edit. Ultimately, we realized that our problem is best approached not as a collision *prevention/resolution*, rather as an existing spacetime mess that must be *untangled*.

5.6.2 Untangling Spacetime Cloth

We extend the untangling cloth method of [Volino and Magnenat-Thalmann, 2006a] to spacetime. Although we consider reconfigurable composed only of *rigid* objects moving in space, the corresponding trajectory spacetime volumes are highly deformable and thus susceptible to “tangling,” analogous to how cloth tangles in $\mathbb{R}^3$.

Volino and Thalmann [2006a] consider the global problem of identifying and minimizing the amount of overlap between deformable cloth surfaces in space. They do so by identifying the one-dimensional intersection contours and minimize their lengths via gradient descent.

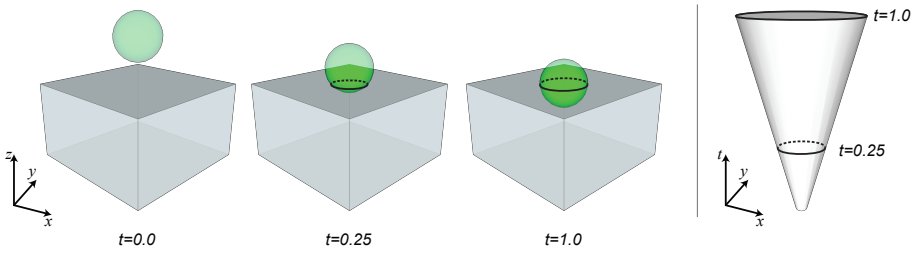


Figure 5.13: Intersection between sphere and box. We illustrate intersection contours for three instants in time (three images, left), and depict the surface swept by all of intersection contours throughout the duration of the intersection (one image, right).

5.6.3 Intersection Surface

At first, it appears that we require a full generalization of this approach from three to four dimensions. Resolving intersections between our spacetime trajectories indeed requires minimizing the *two-dimensional surface area* of the intersection of two three-dimensional volumes in spacetime.

We are saved however by the grace of the monotonicity of time. Time serves as a natural independent variable for parameterizing the spacetime trajectory and obtaining a 3D "slice" at an instant of time. A corollary is that spacetime trajectories may fold over in space but cannot fold over in time (see Figure 5.12). We exploit this crucial fact to simplify the spacetime generalization of the contour minimization method.

Suppose that the time interval $[t_0, t_1]$ encloses a collision between a pair of objects (i, j). At any instance $t \in [t_0, t_1]$, there exists at least one intersection contour (curve) between object i and object j (see Figure 5.13). Integrating these contours over the entire intersection interval sweeps the entire 3D intersection surface immersed in spacetime. Generalizing the earlier work on cloth untangling, we wish to minimize an *energy* measuring the area of this intersection surface $|I_S|$,

$$|I_S| = \int_{t_0}^{t_1} \|I_c(t)\| \mathrm{d}t \tag{5.1}$$

where, $I_c(t)$ is the intersection contour at a time t and $\|\cdot\|$ measures the length of the contour.

5.6.4 Contact Normals

For untangling two-dimensional cloths in $\mathbb{R}^3$, Volino and Thalmann [2006a] use gradient descent to minimize the length of the one-dimensional intersection curve between the two surfaces: they move the n surface mesh vertices $\mathbf{v} \in \mathbb{R}^{n\times 3}$ along the gradient vector $\nabla_{\mathbf{v}}\|I_c\|$. This gradient vector is referred to as the *contact normal*.

We are dealing with "three-dimensional cloths" in $\mathbb{R}^3 \times [0, 1]$. For now, we analogously consider moving spacetime mesh vertices $\mathbf{v} \in \mathbb{R}^3 \times [0, 1]$ by minimizing the intersection volume's *surface area* $|I_S|$ in Equation (5.1), moving along $\nabla_{\mathbf{v}}|I_S|$. Because integration and differentiation commute, this gradient is simply a sum of the gradient of one-dimensional intersection curves for each moment along the collision interval's temporal extent:

$$\nabla_{\mathbf{v}}|I_S| = \int_{t_0}^{t_1} \nabla_{\mathbf{v}}\|I_c(t)\| \mathrm{d}t \tag{5.2}$$

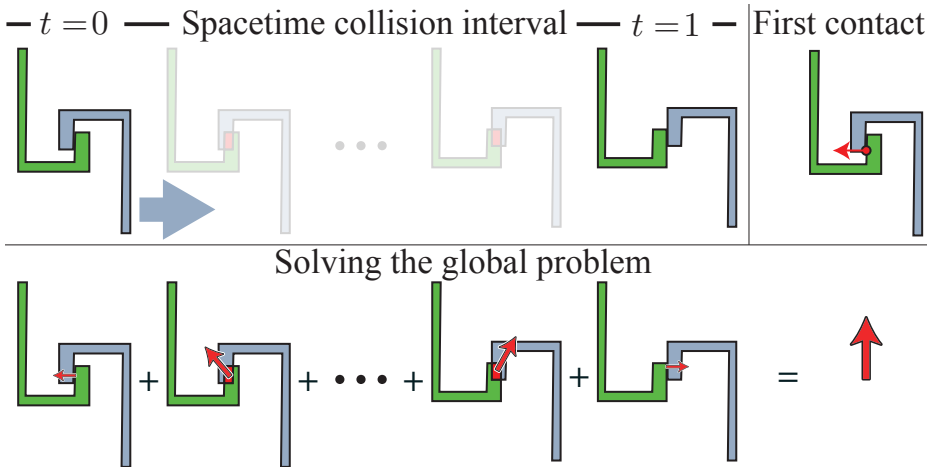


Figure 5.14: Consider a 2D example of a grey hook moving horizontally over a green hook. The spatial normal at first (or last) contact has no vertical component; moving in this direction will only lead to more contact. In contrast, our method effectively solves the *global* problem resulting in a normal pointing in the vertical direction, successfully resolving the contact.

Therefore, we directly employ their gradient calculation sampled regularly in time.

Since each one-dimensional intersection curve I_c is independent of time, the temporal component of our spacetime contact normal is zero. As a concrete example, consider the spatial component of the spacetime contact normal depicted in Figure 5.14. Naturally, the gradient of the *spatial* intersection curve length ($\|I_c(t)\|$) at a fixed time (t) with respect to the mesh vertex *spatial positions* at that time ($\mathbf{v}_t \in \mathbb{R}^{n\times 3}$) only reveals *spatial* information: all slices of the mesh through time seemingly stay in their time slice and do not influence each other.

However, this apparent independence between temporal slices is not as it initially appears. The spacetime mesh is coupled across time via our design user-interface degrees of freedom. We only consider rigid objects, so a change to an object's geometry will affect all moments in time. For example, changing a "rest pose" vertex position $\mathbf{p}_i$ will effect the vertex's position $\mathbf{v}_i(t)$ at every time t. Similarly, the keyframes used for UI elements (cf. §5.4) effect long stretches of time. Generally speaking, the gradient $\nabla_{\text{UI}}|I_S|$ with respect to the UI will have a non-zero temporal action. Letting $\mathbf{G}_t = \nabla_{\mathbf{v}_t}\|I_c(t)\|$ and applying the chain rule, we can determine the gradient of the intersection surface area with respect to the user-interface (cf. [Harmon *et al.*, 2011; Umetani *et al.*, 2011]):

$$\nabla_{\text{UI}}|I_S| = \text{diag}\left((\nabla_{\text{UI}}\mathbf{v}_0)\mathbf{G}_0, \ldots, (\nabla_{\text{UI}}\mathbf{v}_1)\mathbf{G}_1\right), \tag{5.3}$$

that is, a block diagonal matrix involving the gradient of the mesh vertex positions with respect to the UI elements at each discrete time sample, $\nabla_{\text{UI}}\mathbf{v}_t$. Although the temporal component of $\mathbf{G}_t$ is set to zero due to the time independence of $\|I_c(t)\|$, the chain-rule when applied to UI elements results in non-zero changes to the timing of UI keyframes. This follows since the timing of keyframes influence the object's motion in space. We apply the chain rule to this term for our most common UI elements (spherically interpolated quaternion and spline translation keyframes) in the Appendix, application to other user interactions follows analogously.

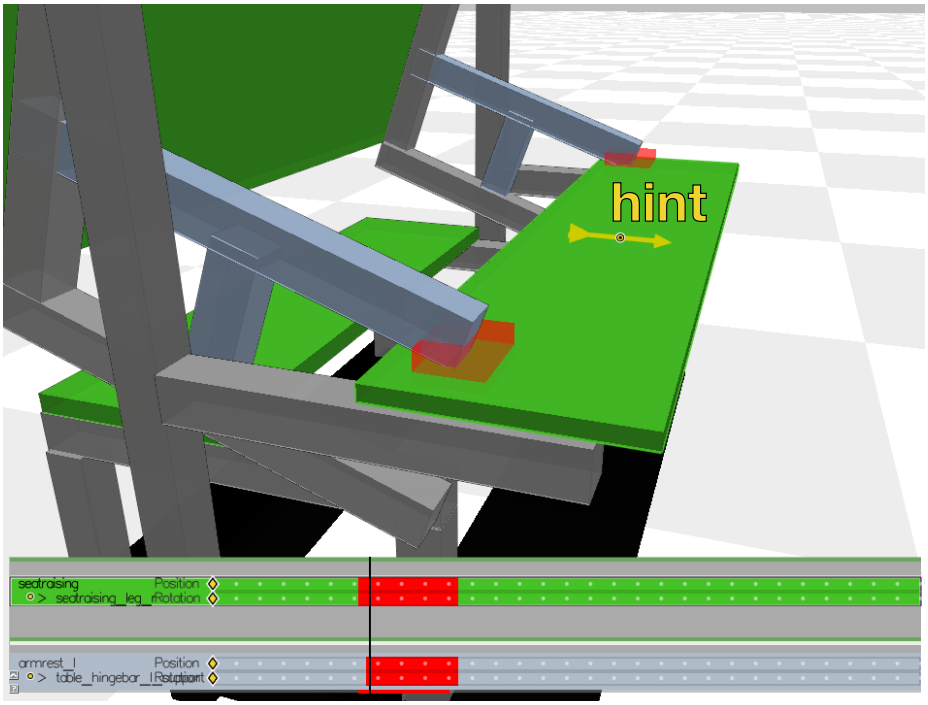


Figure 5.15: The yellow arrow suggests how the designer could move the seat to avoid interference with the armrests.

5.6.5 Resolution Hints

The contact normal computed in §5.6.4 immediately provides useful insight that the designer can harness to resolve interferences. The designer can explicitly request a hint for selected object of the current collision interval (CI).

The chain rule derived in the Appendix effectively projects the energy descent direction on a particular UI element's degrees of freedom, providing exactly the appropriate information necessary to draw a hint. For example, the components corresponding to a spline control point are interpreted as 3-vector in space rendered on screen as a large yellow arrow (see Figure 5.15). The 3D interface can make it difficult to precisely move in the direction of the hint, so we also provide a slider to move the UI element along the hint direction. The interface and the designer work symbiotically to solve the intersection.

Often however this magnitude can be determined automatically and the direction may change as result. This leads naturally to a gradient descent resolution scheme that is fully automatic.

5.6.6 Automatic Resolution

In many instances during the design process the designer is required to make several tedious edits that can often be automatically handled by the system. We formulate this automatic resolution scheme as a non-linear optimization problem that attempts to minimize the energy formulated as the surface area of the collision interface, as described in Equation (5.1). Our implementation of gradient descent employs the Golden Section Search Algorithm [Press *et al.*, 1992b] and assumes that the local energy

landscape has a single global minimum nearby.

Our experiments showed higher success resolving multi-object interferences in pairs independently rather than as a simultaneous optimization over all pairwise interferences. We employ a greedy approach, resolving the collision between the pair of objects contributing most to the intersection energy (5.1). Resolving the most severe collisions first mimics the approach a designer might take, focusing first on severe collisions and later on grazing cases.

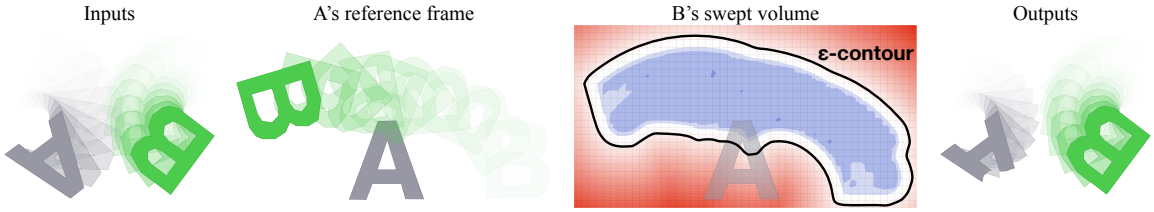


Figure 5.16: Two objects A and B overlap in spacetime (ghosting projection, left). We consider the relative motion of B in A's reference frame. Sampling densely in time we aggregate the signed distance to B on a grid as B transitions. We contour the ϵ-offset surface to the swept volume and subtract this mesh from A. The new A does not overlap B in space time.

5.6.7 Spacetime Geometry Carving

In the previous sections we considered ways to mitigate collisions by altering objects via the available user-interface elements. We now consider altering an object's geometry dramatically by *carving* away the relative swept volume of another interfering object.

Ideally we would like to *subtract*—in a constructive solid geometry sense—an offset of the relative swept volume of the carving object B from the carved object A:

$$A \leftarrow A \setminus \text{offset}\left(\text{sweep}\left(B, f_A^{-1} \circ f_B\right), e\right) \tag{5.4}$$

where $\text{offset}(X, e) \subset \mathbb{R}^3$ computes an e-offset of a given volume $X \subset \mathbb{R}^3$ and $\text{sweep}(X, f) \subset \mathbb{R}^3$ computes a swept volume of a given volume X undergoing a rigid motion $f(t)$.

If B and A are each undergoing rigid motions $f_A(t)$ and $f_B(t)$ respectively, then we consider the swept volume of B according to its motion *observed* in the reference frame of A.

Computing offset volumes and swept volumes of triangle meshes *exactly* poses computational and representational problems. The *exact* swept volume of a solid bounded by a triangle mesh undergoing a rigid motion is a piecewise-ruled surface [Weld and Leu, 1990] (i.e., in general not representable with mesh of flat triangles). Similarly the *exact* offset surface of a solid bounded by a triangle mesh is a piecewise quadric surface [Pavic and Kobbelt, 2008]. However, to fit into the

Figure 5.17: A cluttered kitchen cabinet is tidied up by inserting a styrofoam block and subtracting the swept volumes of each object placed into place. Using a 3D printer, we validate this design.

rest of our pipeline, we need the output of this carving sub-routine to produce a new triangle mesh. Therefore, previous works on exact offsets and swept volumes are inappropriate in our contexts due to their output representation.

Previous tools for computing triangle-mesh approximations of swept volumes (e.g., [Peternell *et al.*, 2005]) and surface offsetting (e.g., [Campen and Kobbelt, 2010]) focus on accuracy over performance and simplicity. Instead, we propose directly computing a conservative triangle-mesh approximation of an offset to the swept volume by contouring a signed distance field. We then subtract this approximation from the exact geometry of the static object to ensure its details remain in tact.

Approximate offset of swept volume Our approach adapts and accelerates the implicit method of [Schroeder *et al.*, 1994]. Without loss of generality, the input to this subroutine is an object B, a rigid motion $f(t)$ for $t \in [0, 1]$, and a desired offset amount e. The output is a triangle mesh approximating the e-offset surface to the swept volume of B moving along $f(t)$ (see Figure 5.16).

First we lay a grid over the bounding box containing the spatial extent of B transformed by $f(t)$ for all $t \in [0, 1]$ padded by $2e$. The grid step size h is an exposed parameter trading off between computation time (larger is coarser, faster) and accuracy (smaller is finer, more accurate). The step size should be chosen smaller than the smallest relevant feature on B.

For each grid vertex, we will approximate the signed distance to the swept volume's surface. The swept volume S is the union of B moving along $f(t)$:

$$S = \bigcup_{t \in [0,1]} f(t)B. \tag{5.5}$$

This is easily re-written in terms of signed distances (assuming negative distances inside a solid):

$$d(S, \mathbf{p}) = \min_{t \in [0,1]} d\left(f(t)B, \mathbf{p}\right), \tag{5.6}$$

where $d(X, \mathbf{p})$ is the signed distance from the surface of some solid X to a query point $\mathbf{p}$.

Given a signed distance field to S, the surface of S (or any offset) can be extracted as the zero (ε) level set. We approximate this on our grid by taking small discrete steps in time and using marching cubes to extract the level set at e. Because the signed-distance field is only useful insofar as it reveals the e level set, we cull grid vertices determined far enough away ($> \sqrt{3}e$) or too far inside ($< e - \sqrt{3}e$) during acceleration tree evaluation.

Finally, we subtract our triangle mesh approximation of S from the triangle mesh representation of A using the robust boolean library within LIBIGL [Jacobson *et al.*, 2013].

It is tempting to conduct this final boolean subtraction also on the signed distance grid, using the signed distance to A. This would certainly be more efficient, but unfortunately forfeits the sharp details and sparse representation of A.

Instead, our approach uses an approximate representation of the swept volume S, but conducts the boolean subtraction exactly. This maintains the original details of A away from the subtracted region, forfeiting details of the swept volume, but this is already abstracted from the designer and obscured by the necessary offsetting.

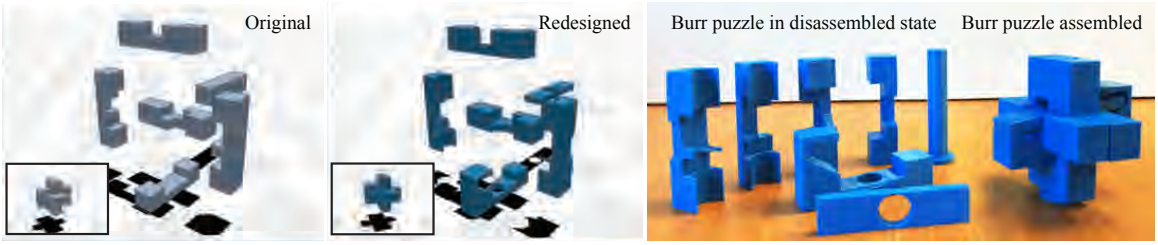


Figure 5.18: We validated the feasibility of our Burr puzzle design by 3D printing the parts.

5.7 Results

We implemented our prototype in C++11 using a background thread (`std::thread`) to detect collisions as the designer makes edits. On a MacBook Air 2GHz Intel Core i7 8GB memory machine, our spacetime collision detection and resolution runs interactively for our examples (meshes with 100 to 10,000 triangles). For examples with multiple transitions, each transition runs an instance of our 3D viewer with keyframe timeline. End states and object geometries are shared and edits propagate immediately. Changes to the collision intervals (CIs) are announced to the graph view.

In Figure 5.2, the designer would like to add armrests to a park bench that reconfigures into a picnic table. The armrests do not cause problems at either end state, but do cause collisions along the way. The designer experiments with a variety of solutions available within our tool: (a) carving away the swept volume of the armrests from the bench seat, (b) interactively reshaping the armrests until collisions disappear, and (c) adding a hinge mechanism to stow the armrests beneath the table.

In Figure 5.4, the designer adds various accessories to a folding bicycle. Adding a basket and altering the handle bars creates collisions (red highlight) when folding laterally. After interactive experimentation, the designer finds that changing the folding mechanism will accommodate the additions.

The reconfigurable kitchen in Figure 5.1 involves a graph of six transitions between seven states. We show a few of the most interesting states with ghosting (see §5.4.2) to indicate transitions. In the clean up state (1), all deployable objects are hidden and there is plenty of space to walk around. In the food prep state (2), the stove vent lowers to reveal more counter space. In appliance state (4), medium-size machines appear ready for use via a Murphy-bed-style shelving system. If more counter space is needed when using the appliances, extra counter space unfolds over the sink (5). The designer employs the carving tool of §5.6.7 to leave space for the faucet. When cooking is done and the appliances are put away, the telescoping table deploys and benches swing out from under the cabinets (6). See the accompanying material for a video of the full length editing session of this example.

Reconfigurability helps tidy up a chaotic cabinet of appliances, cups, and glass in Figure 5.17. In this theoretical example, the designer considers filling the cabinet with a block of styrofoam and then carving from it the swept volume of each object as it is placed into a non-overlapping position in the cabinet. We validated the effectiveness of this idea with a 3D-printed scale model.

In Figure 5.19, the designer plans furniture arrangements for a reconfigurable studio apartment in Manhattan. By identifying and resolving impossible transitions, the apartment fits a wide assortment of furniture featuring a sleeping mode, bathroom mode, and entertainment mode.

Our tools function on the macro apartment-size scale and also on the micro scale. In Figure 5.18,

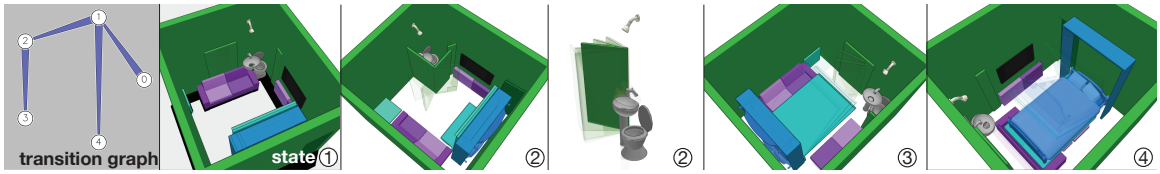


Figure 5.19: A reconfigurable apartment. Transition graph (leftmost) summarizes four overall states. (1) An open floorplan is used for parties and housekeeping. (2) In daytime and evening, the sofa faces the TV, the WC is accessible via a swinging door, and for showering, the sink folds up, the undersink piping telescopes into the drain. (3) For dining, a table and benches swing into place; the sink remains accessible but not WC. (4) At night, a wall bed swings down.

the designer reshapes and adds complexity to a reconfigurable Burr puzzle. We validated this result using a 3D printer: all parts fit together neatly.

5.7.1 Limitations & Future Work

We focused the feature set of our prototype on visualization, monitoring and resolution aids unique to the problem of designing reconfigurables. We delegated advanced geometric editing to third party tools. Integrating advanced real-time mesh editing [Gal *et al.*, 2009b; Liu *et al.*, 2014] into our tool should be possible.

We also assumed that the designer had geometric models available. While 3D scanning is becoming more common place, we imagine that integrated the wealth of online 3D data (à la [Schulz *et al.*, 2014a]) would be an interesting extension.

Finally, our interpretation of physical feasibility is limited to interpenetration during reconfiguration. We rely on the user's human intuition or domain expertise to prevent unnatural or mechanically impossible transitions (e.g., levitating couches). Adapting our contact resolution optimization to account for mechanical constraints would be a challenging but exciting future work.

Chapter 6

Conclusions

The work presented in this thesis is concerned with finding efficient computational models that alleviate the user from the overwhelming task of maintaining external constraints on the shape or environment. The work has resulted in three distinct but related publications towards this goal: [Garg *et al.*, 2007; Garg *et al.*, 2014; Garg *et al.*, 2016]. We have seen that certain geometric insights allow us to create the necessary abstractions for efficient computation that assist design interfaces. The goal has been to have computation run in the background while providing the human tools of control and expressiveness.

In *Cubic Shells* (Chapter 3) we saw that deformations restricted to isometry can lead to an efficient computational model for simulating thin shells. Modeling thin shells this way also helped to create an expressive but simple model for capturing and manipulating orthotropy in the material by simply "painting" a texture map.

Wire Meshes (Chapter 4) have a similar geometric foundation as many thin shells, except there we exploit the mathematical theory laid out by Hazzidakis in order to understand the underlying fundamental limitations of the design space. If we naively attempt to cover an arbitrary surface with some wire mesh we will fail miserably. Hazzidakis's work helps us to understand this and directed us to an insight that allows our designs to deviate slightly from the underlying surface. The target surface in this workflow remains as a guide instead of a shape constraint that we must interpolate. This provides immense flexibility in developing our user tools that create expressive freedom during the design process.

Finally, to truly show a symbiotic relationship between computational and human control we consider the design space of *Reconfigurables* (Chapter 5). Here, we explored a subset of interactive tools necessary for a typical user to reason about complicated state transitions between different configurations. All the while, our computational machinery is able to guide the user when transitions/configurations become invalid due to erroneous collisions, draw the attention of the user to where and when those occur, as well as provide potential solutions to fix those collisions.

The work in this thesis focuses on a subset of problem domains that could benefit from a computationally driven design paradigm, providing more control and expressiveness, inevitably leading to creative inspiration. We can continue to push the limits of the quality and control we can achieve with our designs as: (i) our computational machinery becomes more ubiquitously available (ii) we become more adept to working with and adopting new technological innovations, and (iii) there is massive reduction in the time it takes to realize our designs with on-demand visualization and fabrication. There is tremendous opportunity for the future of computationally assisted design.

Usability Computer simulations generally have too many parameters that are too hard to control. As we introduce the human into the design loop we must develop models that rely on far less and far fewer parameters. This is especially true for computationally heavy methods that require the use of background simulations in order to explore the design space. For example, our work presented in Chapter 3 alleviates the user from configuring parameters for orthotropic materials and instead simply expects a scalar field which can be embedded in a texture map. Understanding these parameters will also allow us to amplify the necessary ones to expose richer behavior. Recent efforts at interactive design, sketching, and novel tactile input devices show promise in this direction. Furthermore, it can often be challenging to determine what information is pertinent to the user. A preview of this was explored in our work in Chapter 5, but further research in this direction could benefit design tools by refining view regions that are optimized towards context driven collision resolution. Finally, usability tools don't have to be primarily driven by software. The work of Jacobson et. al. [2014] and Glauser et. al. [2016] showcases advancements in new input devices composed of embedded, interchangeable parts that allow the user to pose, animate, and interact with digital models in an easy to use intuitive manner.

Complexity and Representation Our appetite for richer, higher quality, and higher resolution geometry is increasing. Furthermore, as digital representations of designs and objects increase so do their interrelationships. Dealing with complex interactions and constraints between many objects remains an open problem. While there are various different design techniques in dealing with different sets of constraints, many methods will augment each other. Each of these methods will have their own set of representations that must play with each other nicely and potentially at multiple scales. Put simply, choosing, storing, managing, and integrating different representations will be challenging as industries move towards digitized pipelines for design and fabrication. For example, in the context of reconfigurables (Chapter 5), the burden currently lies on the user to determine feasible mechanical transitions and constructions. Ideally, the user is able to draw from datasets that define feasible relationships between objects that adhere to certain limitations and constraints – these constraints/designs could then serve to provide the mechanical underpinnings necessary to validate the the user's designs remain feasible as well as collision free during transitions. The rise in the availability of digital models can also give rise to more data-driven design methods, such as the one presented by Schulz et. al. [2014b]. Their work show how large datasets can be used in the context of interactive design by using available templates, changing their parameters and combining them with other parts to create new and novel designs.

Cognitive Modeling and AI Whether due to increasing complexity or to the need for modularization and autonomy, we will require abstractions that can encapsulate complex masses of data and higher-levels of reasoning. Artificial intelligence seems to be trendy these days, but advancements in that field can certainly be used to assist in managing complex interactive systems. These cognizant systems can be used to teach, document, predict and suggest tasks that normally would require expert users. Such methods would be especially useful in computational design tools that require the user to make trade-offs. For wire mesh design (Chapter 4), the user must often decide between surface coverage and guide surface adherence. Although artistic intuition plays a leading role here, a cognizant system could provide suggestions not only about other valid decisions, but also about any additional constraints that the current trade off has imposed on the design. Assistive predictions and suggestions can also be useful in the context of providing globally optimal collision free designs, as

discussed in Chapter 5. Our current algorithm uses a variant of gradient descent that is prone to get stuck in a local minima and loses context over the user's design intent. High level constraints and globally optimal suggestions/solutions can be feasible if the system is able to "learn" from a set of training examples.

We hope that the work presented in this thesis will inspire researchers to explore and extend beyond the humble ideas presented here.

Chapter 7

Bibliography

Bibliography

[Adkins, 1956] J E Adkins. Finite Plane Deformation of Thin Elastic Sheets Reinforced with Inextensible Cords. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 249(961):125–150, 1956.

[Allard *et al.*, 2010] Jérémie Allard, Franccois Faure, Hadrien Courtecuisse, Florent Falipou, Christian Duriez, and Paul G. Kry. Volume contact constraints at arbitrary resolution. *ACM Trans. Graph.*, 29(4):82:1–82:10, July 2010.

[An *et al.*, 2008] Steven S. An, Theodore Kim, and Doug L. James. Optimizing cubature for efficient integration of subspace deformations. In *ACM SIGGRAPH Asia 2008 Papers*, SIGGRAPH Asia '08, pages 165:1–165:10, New York, NY, USA, 2008. ACM.

[Aono *et al.*, 1996] M Aono, P Denti, D E Breen, and M J Wozny. Fitting a woven cloth model to a curved surface: dart insertion. *IEEE Computer Graphics and Applications*, 16(5):60–70, 1996.

[Aono *et al.*, 2001] M Aono, D E Breen, and M J Wozny. Modeling methods for the design of 3D broadcloth composite parts. *Computer-Aided Design*, 33(13):989–1007, 2001.

[Aono, 1994] M Aono. *Computer-Aided Geometric Design for Forming Woven Cloth Composites*. PhD thesis, Rensselaer Polytechnic Institute, January 1994.

[Bächer *et al.*, 2012] Moritz Bächer, Bernd Bickel, Doug L. James, and Hanspeter Pfister. Fabricating articulated characters from skinned meshes. *ACM Trans. Graph.*, 2012.

[Bächer *et al.*, 2014] Moritz Bächer, Emily Whiting, Bernd Bickel, and Olga Sorkine-Hornung. Spin-it: Optimizing moment of inertia for spinnable objects. *ACM Trans. Graph.*, 2014.

[Bächer *et al.*, 2015] Moritz Bächer, Stelian Coros, and Bernhard Thomaszewski. Linkedit: interactive linkage editing using symbolic kinematics. *ACM Trans. Graph.*, 2015.

[Baillargeon and Vu-Khanh, 2001] Y Baillargeon and T Vu-Khanh. Prediction of fiber orientation and microstructure of woven fabric composites after forming. *Composite Structures*, 52(3-4):475–481, 2001.

[Bakelman, 1965] Ilya Yakovlevich Bakelman. Chebyshev networks in manifolds of bounded curvature. *Trudy Matematicheskogo Instituta im. VA Steklova*, 76:124–129, 1965.

[Baraff and Witkin, 1998] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *SIGGRAPH*, pages 43–54. ACM Press, 1998.

[Baraff *et al.*, 2003] D. Baraff, A. Witkin, and M. Kass. Untangling cloth. *ACM Trans. Graph.*, 22(3):862–870, 2003.

[Barbič and James, 2005] Jernej Barbič and Doug L James. Real-time subspace integration for st. venant-kirchhoff deformable models. In *ACM transactions on graphics (TOG)*, volume 24, pages 982–990. ACM, 2005.

[Bergou *et al.*, 2006] Miklos Bergou, Max Wardetzky, David Harmon, Denis Zorin, and Eitan Grinspun. A quadratic bending model for inextensible surfaces. In *SGP*, pages 227–230, 2006.

[Bernstein and Wojtan, 2013] Gilbert Louis Bernstein and Chris Wojtan. Putting holes in holey geometry: Topology change for arbitrary surfaces. *ACM Trans. Graph.*, 2013.

[Bharaj *et al.*, 2015] Gaurav Bharaj, David I. W. Levin, James Tompkin, Yun Fei, Hanspeter Pfister, Wojciech Matusik, and Changxi Zheng. Computational design of metallophone contact sounds. *ACM Trans. Graph.*, 2015.

[Bieberbach, 1926] Ludwig Bieberbach. über Tchebychefsche Netze auf Flächen negativer krümmung, sowie auf einigen weiteren flächenarten. *Preuss. Akad. Wiss., Phys. Math. Kl*, 23:294–321, 1926.

[Bo *et al.*, 2011] Pengbo Bo, Helmut Pottmann, Martin Kilian, Wenping Wang, and Johannes Wallner. Circular arc structures. *ACM Trans. Graph. (SIGGRAPH '11)*, 30(4):101:1–101:11, 2011.

[Bobenko and Pinkall, 1996] Alexander I Bobenko and Ulrich Pinkall. Discrete surfaces with constant negative Gaussian curvature and the Hirota equation. *Journal of Differential Geometry*, 43(3):527–611, 1996.

[Bokeloh *et al.*, 2010] Martin Bokeloh, Michael Wand, and Hans-Peter Seidel. A connection between partial symmetry and inverse procedural modeling. *ACM Trans. Graph.*, 29(4):104:1–104:10, July 2010.

[Bokeloh *et al.*, 2011] Martin Bokeloh, Michael Wand, Vladlen Koltun, and Hans-Peter Seidel. Pattern-aware shape deformation using sliding dockers. *ACM Trans. Graph.*, 30(6):123:1–123:10, December 2011.

[Bokeloh *et al.*, 2012] Martin Bokeloh, Michael Wand, Hans-Peter Seidel, and Vladlen Koltun. An algebraic model for parameterized shape editing. *ACM Trans. Graph.*, 31(4):78:1–78:10, July 2012.

[Bouaziz *et al.*, 2012] Sofien Bouaziz, Mario Deuss, Yuliy Schwartzburg, Thibaut Weise, and Mark Pauly. Shape-up: Shaping discrete geometry with projections. *Comp. Graph. Forum (SGP '12)*, 31(5):1657–1667, 2012.

[Breen and Donald, 1994] D.E. Breen and H. Donald. House, and Michael J. Wozny. Predicting the drape of woven cloth using interacting particles. *Proceedings of SIGGRAPH '94*, pages 365–372, 1994.

[Bridson *et al.*, 2003] Robert Bridson, Sebastian Marino, and Ronald Fedkiw. Simulation of clothing with folds and wrinkles. *SCA*, pages 28–36, 2003.

[Burago *et al.*, 2007] Y D Burago, S V Ivanov, and S G Malev. Remarks on Chebyshev Coordinates. *Journal of Mathematical Sciences*, 140(4):497–501, 2007.

[Buxton, 2007] Bill Buxton. *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[Cameron, 1990] Stephen Cameron. Collision detection by four-dimensional intersection testing. *IEEE T. Robotics and Automation*, 1990.

[Campen and Kobbelt, 2010] Marcel Campen and Leif Kobbelt. Polygonal boundary evaluation of minkowski sums and swept volumes. *Comput. Graph. Forum*, 2010.

[Ceylan *et al.*, 2013] Duygu Ceylan, Wilmot Li, Niloy J. Mitra, Maneesh Agrawala, and Mark Pauly. Designing and fabricating mechanical automata from mocap sequences. *ACM Trans. Graph.*, 2013.

[Choi and Ko, 2003] K.-J. Choi and H.-S. Ko. Extending the immediate buckling model to triangular meshes for simulating complex clothes. In *Eurographics 2003 Short Presentations*, pages 187–191, 2003.

[Cignoni *et al.*, 2014] Paolo Cignoni, Nico Pietroni, Luigi Malomo, and Roberto Scopigno. Field-aligned mesh joinery. *ACM Trans. Graph.*, 33(1):11:1–11:12, 2014.

[Cirio *et al.*, 2014] Gabriel Cirio, Jorge Lopez-Moreno, David Miraut, and Miguel A Otaduy. Yarn-level simulation of woven cloth. *ACM Transactions on Graphics (TOG)*, 33(6):207, 2014.

[Coros *et al.*, 2013] Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert W. Sumner, Wojciech Matusik, and Bernd Bickel. Computational design of mechanical characters. *ACM Trans. Graph.*, 32(4):83:1–83:12, July 2013.

[de Goes *et al.*, 2013] Fernando de Goes, Pierre Alliez, Houman Owhadi, and Mathieu Desbrun. On the equilibrium of simplicial masonry structures. *ACM Trans. Graph. (SIGGRAPH '13)*, 32(4):93:1–93:10, 2013.

[Deng *et al.*, 2013] Bailin Deng, Sofien Bouaziz, Mario Deuss, Juyong Zhang, Yuliy Schwartzburg, and Mark Pauly. Exploring local modifications for constrained meshes. *Computer Graphics Forum (EUROGRAPHICS '13)*, 32(2):11–20, 2013.

[Deng *et al.*, 2014] Bailin Deng, Sofien Bouaziz, Mario Deuss, Alexandre Kaspar, Yuliy Schwartzburg, and Mark Pauly. Interactive design exploration for constrained meshes. *Computer-Aided Design*, 2014. To appear.

[do Carmo, 1992] Manfredo do Carmo. *Riemannian Geometry*. Mathematics: Theory and Applications. Birkhäuser, 1992.

[Dodgson *et al.*, 2005] Mark Dodgson, David Gann, and Ammon J Salter. *Think, play, do: Technology, innovation, and organization*. Oxford University Press on Demand, 2005.

[Doherty and Thadhani, 1982] Walter J Doherty and Arvind J Thadhani. The economic value of rapid response time. *IBM Report*, 1982.

[Eigensatz *et al.*, 2010] Michael Eigensatz, Martin Kilian, Alexander Schiftner, Niloy Mitra, Helmut Pottmann, and Mark Pauly. Paneling architectural freeform surfaces. *ACM Trans. Graph. (SIGGRAPH '10)*, 29(4):45:1–45:10, 2010.

[Erleben, 2004] Ken Erleben. *Stable, Robust, and Versatile Multibody Dynamics Animation*. PhD thesis, Univ. of Copenhagen, 2004.

[Everitt, 2001] Cass Everitt. Interactive order-independent transparency. Technical report, nVidia Corp., 2001.

[Feng *et al.*, 2010] Wei-Wen Feng, Yizhou Yu, and Byung-Uck Kim. A deformation transformer for real-time cloth animation. In *ACM Transactions on Graphics (TOG)*, volume 29, page 108. ACM, 2010.

[Fisher *et al.*, 2011] Matthew Fisher, Manolis Savva, and Pat Hanrahan. Characterizing structural relationships in scenes using graph kernels. *ACM Trans. Graph.*, 30(4):34:1–34:12, July 2011.

[Gal *et al.*, 2009a] Ran Gal, Olga Sorkine, Niloy J. Mitra, and Daniel Cohen-Or. iwires: An analyze-and-edit approach to shape manipulation. *ACM Transactions on Graphics (Siggraph)*, 28(3):#33, 1–10, 2009.

[Gal *et al.*, 2009b] Ran Gal, Olga Sorkine, Niloy J. Mitra, and Daniel Cohen-Or. iWIRES: An analyze-and-edit approach to shape manipulation. *ACM Trans. Graph.*, 2009.

[Garg *et al.*, 2007] A. Garg, E. Grinspun, M. Wardetzky, and D. Zorin. Cubic shells. *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 91–98, 2007.

[Garg *et al.*, 2014] Akash Garg, Andrew O. Sageman-Furnas, Bailin Deng, Yonghao Yue, Eitan Grinspun, Mark Pauly, and Max Wardetzky. Wire mesh design. *ACM Trans. Graph.*, 2014.

[Garg *et al.*, 2016] Akash Garg, Alec Jacobson, and Eitan Grinspun. Computational design of reconfigurables. *ACM Trans. Graph.*, 2016.

[Ghys, 2011] Étienne Ghys. Sur la coupe des vêtements: variation autour d'un thème de Tchebychev. *L'Enseignement Mathématique Revue Internationale 2e Sèrie*, 57(1-2):165–208, 2011.

[Gingold *et al.*, 2004] Yotam Gingold, Adrian Secord, Jefferson Y. Han, Eitan Grinspun, and Denis Zorin. A Discrete Model for Inelastic Deformation of Thin Shells. Technical report, Aug 2004.

[Glauser *et al.*, 2016] Oliver Glauser, Wan-Chun Ma, Daniele Panozzo, Alec Jacobson, Otmar Hilliges, and Olga Sorkine-Hornung. Rig animation with a tangible and modular input device. *ACM Transactions on Graphics (TOG)*, 35(4):144, 2016.

[Grinspun *et al.*, 2003] Eitan Grinspun, Anil N. Hirani, Mathieu Desbrun, and Peter Schröder. Discrete shells. *SCA*, pages 62–67, 2003.

[Grinspun *et al.*, 2006] Eitan Grinspun, Yotam Gingold, Jason Reisman, and Denis Zorin. Computing discrete shape operators on general meshes. *Comput. Graph. Forum*, 25(3), 2006.

[Guibas, 1998] Leonidas J Guibas. Kinetic data structures–a state of the art report. *Proc. WAFR*, 1998.

[Hairer *et al.*, 2006] E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer, 2006.

[Harmon and Zorin, 2013] David Harmon and Denis Zorin. Subspace integration with local deformations. *ACM Transactions on Graphics (TOG)*, 32(4):107, 2013.

[Harmon *et al.*, 2011] David Harmon, Daniele Panozzo, Olga Sorkine, and Denis Zorin. Interference-aware geometric modeling. *ACM Trans. Graph.*, 2011.

[Hauschild and Karzel, 2011] Moritz Hauschild and Rüdiger Karzel. *Digital processes: planning, designing, production*. Walter de Gruyter, 2011.

[Hauth, 2004] Michael Hauth. *Visual Simulation of Deformable Models*. PhD thesis, University of Tübingen, 2004.

[Hazzidakis, 1879] J N Hazzidakis. Über einige Eigenschaften der Flächen mit constantem Krümmungsmass. *Journal für die reine und angewandte Mathematik*, 88:68–73, 1879.

[Hearle *et al.*, 2001] JWS Hearle, P Potluri, and VS Thammandra. Modelling fabric mechanics. *Journal of the Textile Institute*, 92(3):53–69, 2001.

[Hildebrandt and Polthier, 2004] Klaus Hildebrandt and Konrad Polthier. Anisotropic filtering of non-linear surface features. *CGF*, 23(3):391–400, 2004.

[Hoffmann, 1999] Tim Hoffmann. Discrete Amsler surfaces and a discrete Painlevé III equation. *Oxford Lecture Series in Mathematics and its Applications*, 16:83–96, 1999.

[Igarashi *et al.*, 1999] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: A sketching interface for 3d freeform design. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 409–416, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[Igarashi *et al.*, 2012] Yuki Igarashi, Takeo Igarashi, and Jun Mitani. Beady: Interactive beadwork design and construction. *ACM Trans. Graph.*, 2012.

[Jacobson *et al.*, 2013] Alec Jacobson, Daniele Panozzo, et al. libigl: A simple C++ geometry processing library, 2013. http://igl.ethz.ch/projects/libigl/.

[Jacobson *et al.*, 2014] Alec Jacobson, Daniele Panozzo, Oliver Glauser, Cédric Pradalier, Otmar Hilliges, and Olga Sorkine-Hornung. Tangible and modular input device for character articulation. *ACM Transactions on Graphics (TOG)*, 33(4):82, 2014.

[Jain *et al.*, 2012] Arjun Jain, Thorsten Thormählen, Tobias Ritschel, and Hans-Peter Seidel. Exploring shape variations by 3d-model decomposition and part-based recombination. *Computer Graphics Forum*, 31(2pt3):631–640, 2012.

[James and Fatahalian, 2003] Doug L James and Kayvon Fatahalian. *Precomputing interactive dynamic deformable scenes*, volume 22. ACM, 2003.

[Joubert *et al.*, 2015] Niels Joubert, Mike Roberts, Anh Truong, Floraine Berthouzoz, and Pat Hanrahan. An interactive tool for designing quadrotor camera shots. *ACM Trans. Graph.*, 2015.

[Kalojanov *et al.*, 2012] Javor Kalojanov, Martin Bokeloh, Michael Wand, Leonidas Guibas, Hans-Peter Seidel, and Philipp Slusallek. Microtiles: Extracting Building Blocks from Correspondences. *Computer Graphics Forum*, 31(5):1597–1606, 2012.

[Kavan *et al.*, 2011] Ladislav Kavan, Dan Gerszewski, Adam W Bargteil, and Peter-Pike Sloan. Physics-inspired upsampling for cloth simulation in games. In *ACM Transactions on Graphics (TOG)*, volume 30, page 93. ACM, 2011.

[Kavraki *et al.*, 1996] Lydia E Kavraki, Petr Švestka, Jean-Claude Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE TRA*, 1996.

[Kawabata, 1980] S. Kawabata. *The Standardization and Analysis of Hand Evaluation*. The Hand Evaluation and Standardization Committee, The Textile Machinery Society of Japan, 1980.

[Kim *et al.*, 2013] Doyub Kim, Woojong Koh, Rahul Narain, Kayvon Fatahalian, Adrien Treuille, and James F O'Brien. Near-exhaustive precomputation of secondary cloth effects. *ACM Transactions on Graphics (TOG)*, 32(4):87, 2013.

[Klosowski *et al.*, 1998] James T Klosowski, Martin Held, Joseph SB Mitchell, Henry Sowizral, and Karel Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE TVCG*, 1998.

[Koo *et al.*, 2014] Bongjin Koo, Wilmot Li, JiaXian Yao, Maneesh Agrawala, and Niloy J. Mitra. Creating works-like prototypes of mechanical objects. *ACM Trans. Graph.*, 2014.

[Li *et al.*, 2012] Honghua Li, Ibraheem Alhashim, Hao Zhang, Ariel Shamir, and Daniel Cohen-Or. Stackabilization. *ACM Trans. Graph.*, 2012.

[Liu *et al.*, 2006] Yang Liu, Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, and Wenping Wang. Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graph. (SIGGRAPH '06)*, 25(3):681–689, 2006.

[Liu *et al.*, 2013] Yang Liu, Hao Pan, John Snyder, Wenping Wang, and Baining Guo. Computing self-supporting surfaces by regular triangulation. *ACM Trans. Graph. (SIGGRAPH '13)*, 32(4):92:1–92:10, 2013.

[Liu *et al.*, 2014] Songrun Liu, Alec Jacobson, and Yotam Gingold. Skinning cubic Bézier splines and Catmull-Clark subdivision surfaces. *ACM Trans. Graph.*, 2014.

[Liu *et al.*, 2015] Tianqiang Liu, Aaron Hertzmann, Wilmot Li, and Thomas Funkhouser. Style compatibility for 3D furniture models. *ACM Trans. Graph.*, 2015.

[Llach, 2015] Daniel Cardoso Llach. *Builders of the Vision: Software and the Imagination of Design*. Routledge, 2015.

[Löwgren, 2006] Jonas Löwgren. Articulating the use qualities of digital designs. *Fishwick, P. (ed.) Aesthetic computing*, pages 383–404, 2006.

[Mei and Vernescu, 2010] Chiang C Mei and Bogdan Vernescu. *Homogenization methods for multiscale mechanics*. World scientific, 2010.

[Merrell *et al.*, 2011a] Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. Interactive furniture layout using interior design guidelines. *ACM Trans. Graph.*, 30(4):87:1–87:10, July 2011.

[Merrell *et al.*, 2011b] Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. Interactive furniture layout using interior design guidelines. *ACM Trans. Graph.*, 2011.

[Mitra and Pauly, 2008] N. J. Mitra and M. Pauly. Symmetry for architectural design. In *Advances in Architectural Geometry*, pages 13–16, 2008.

[Mitra *et al.*, 2007] Niloy J. Mitra, Leonidas Guibas, and Mark Pauly. Symmetrization. *ACM Transactions on Graphics (SIGGRAPH)*, 26(3):#63, 1–8, 2007.

[Mitra *et al.*, 2010] Niloy J. Mitra, Yong-Liang Yang, Dong-Ming Yan, Wilmot Li, and Maneesh Agrawala. Illustrating how mechanical assemblies work. *ACM Transactions on Graphics*, 29(3):58:1–58:12, 2010.

[Mitra *et al.*, 2012] Niloy J. Mitra, Mark Pauly, Michael Wand, and Duygu Ceylan. Symmetry in 3d geometry: Extraction and applications. In *EUROGRAPHICS State-of-the-art Report*, 2012.

[Möbius and Kobbelt, 2012] Jan Möbius and Leif Kobbelt. Openflipper: An open source geometry processing and rendering framework. In Jean-Daniel Boissonnat, Patrick Chenin, Albert Cohen, Christian Gout, Tom Lyche, Marie-Laurence Mazure, and Larry Schumaker, editors, *Curves and Surfaces*, volume 6920 of *Lecture Notes in Computer Science*, pages 488–500. Springer Berlin Heidelberg, 2012.

[Molino *et al.*, 2004] Neil Molino, Zhaosheng Bao, and Ron Fedkiw. A virtual node algorithm for changing mesh topology during simulation. In *SIGGRAPH*, pages 385–392, 2004.

[Morini, 1999] Benedetta Morini. Convergence behaviour of inexact newton methods. *Math. of Comp.*, 68(228):1605–1613, 1999.

[Nadler *et al.*, 2006] Ben Nadler, Panayiotis Papadopoulos, and David J Steigmann. Multiscale constitutive modeling and numerical simulation of fabric material. *International Journal of Solids and Structures*, 43(2):206–221, 2006.

[O'Brien and Hodgins, 1999] James F. O'Brien and Jessica K. Hodgins. Graphical modeling and animation of brittle fracture. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 137–146, August 1999.

[Oleinik, 1984] OA Oleinik. On homogenization problems. In *Trends and applications of pure mathematics to mechanics*, pages 248–272. Springer, 1984.

[Panozzo *et al.*, 2013] Daniele Panozzo, Philippe Block, and Olga Sorkine-Hornung. Designing unreinforced masonry models. *ACM Trans. Graph. (SIGGRAPH '13)*, 32(4):91:1–91:12, 2013.

[Parsons *et al.*, 2010] Ethan M Parsons, Tusit Weerasooriya, Sai Sarva, and Simona Socrate. Impact of woven fabric: Experiments and mesostructure-based continuum-level simulations. *Journal of the Mechanics and Physics of Solids*, 58(11):1995–2021, 2010.

[Pavic and Kobbelt, 2008] Darko Pavic and Leif Kobbelt. High-resolution volumetric computation of offset surfaces with feature preservation. *Comput. Graph. Forum*, 2008.

[Peternell *et al.*, 2005] Martin Peternell, Helmut Pottmann, Tibor Steiner, and Hongkai Zhao. Swept volumes. *Computer-Aided Design Appl.*, 2005.

[Pinkall, 2008] Ulrich Pinkall. Designing cylinders with constant negative curvature. In *Discrete Differential Geometry*, pages 57–66. Springer, 2008.

[Pipkin, 1984] A C Pipkin. Equilibrium of Tchebychev nets. *Archive for Rational Mechanics and Analysis*, 85(1):81–97, 1984.

[Pipkin, 1986] A C Pipkin. Continuously distributed wrinkles in fabrics. *Archive for Rational Mechanics and Analysis*, 95(2):93–115, 1986.

[Popović *et al.*, 2000] Jovan Popović, Steven M. Seitz, Michael Erdmann, Zoran Popović, and Andrew Witkin. Interactive manipulation of rigid body simulations. In *Proc. SIGGRAPH*, 2000.

[Poranne *et al.*, 2013] Roi Poranne, Elena Ovreiu, and Craig Gotsman. Interactive planarization and optimization of 3D meshes. *Computer Graphics Forum*, 32(1):152–163, 2013.

[Pottmann *et al.*, 2008] H. Pottmann, A. Schiftner, P. Bo, H. Schmiedhofer, W. Wang, N. Baldassini, and J. Wallner. Freeform surfaces from single curved panels. *ACM Trans. Graph. (SIGGRAPH '08)*, 27(3):76:1–76:10, 2008.

[Pottmann *et al.*, 2010] Helmut Pottmann, Qixing Huang, Bailin Deng, Alexander Schiftner, Martin Kilian, Leonidas Guibas, and Johannes Wallner. Geodesic patterns. *ACM Trans. Graph. (SIGGRAPH '10)*, 29(4):43:1–43:10, 2010.

[Press *et al.*, 1992a] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cam. Univ. Press, 1992.

[Press *et al.*, 1992b] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.

[Prévost *et al.*, 2013] Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. Make It Stand: Balancing shapes for 3D fabrication. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, 32(4):81:1–81:10, 2013.

[Rivlin, 1958] R S Rivlin. The deformation of a membrane formed by inextensible cords. *Archive for Rational Mechanics and Analysis*, 2(1):447–476, 1958.

[Rivlin, 1964] R S Rivlin. Networks of inextensible cords. In *Nonlinear Problems of Engineering*, pages 51–64. Academic Press Professional, Inc, New York, 1964.

[Rivlin, 1997] R S Rivlin. Plane Strain of a Net Formed by Inextensible Cords. In Grigori Isaakovich Barenblatt and Daniel D Joseph, editors, *Collected Papers of R.S. Rivlin*, pages 511–534. Springer New York, 1997.

[Robertson *et al.*, 1981] R E Robertson, E S Hsiue, E N Sickafus, and G S Y Yeh. Fiber rearrangements during the molding of continuous fiber composites. I. Flat cloth to a hemisphere. *Polymer composites*, 2(3):126–131, 1981.

[Robertson *et al.*, 2000] R E Robertson, T J Chu, R J Gerard, J H Kim, M Park, H G Kim, and R C Peterson. Three-dimensional fiber reinforcement shapes obtainable from flat, bidirectional fabrics without wrinkling or cutting. Part 2: a single n-sided pyramid, cone, or round box. *Composites Part A: Applied Science and Manufacturing*, 31(11):1149–1165, 2000.

[Saito, 2013] Shunsuke Saito. Microscopic deformation effects in up-sampled cloth simulation. *MI lecture note series*, 50:27–32, 2013.

[Samelson and Dayawansa, 1995] S L Samelson and W P Dayawansa. On the Existence of Global Tchebychev Nets. *Transactions of the American Mathematical Society*, 347(2):651–660, 1995.

[Samelson, 1991] S L Samelson. Global Tchebychev Nets on Complete Two-Dimensional Riemannian Surfaces. *Archive for Rational Mechanics and Analysis*, 114(3):237–254, 1991.

[Sauer, 1970] R Sauer. *Differenzengeometrie*. Springer Verlag, Berlin, 1970.

[Schroeder *et al.*, 1994] William J. Schroeder, William E. Lorensen, and Steve Linthicum. Implicit modeling of swept surfaces and volumes. In *Proc. of the Conference on Visualization*, 1994.

[Schüller *et al.*, 2014] Christian Schüller, Daniele Panozzo, and Olga Sorkine-Hornung. Appearance-mimicking surfaces. *ACM Trans. Graph.*, 2014.

[Schulz *et al.*, 2014a] Adriana Schulz, Ariel Shamir, David I. W. Levin, Pitchaya Sitthi-Amorn, and Wojciech Matusik. Design and fabrication by example. *ACM Trans. Graph.*, 2014.

[Schulz *et al.*, 2014b] Adriana Schulz, Ariel Shamir, David I. W. Levin, Pitchaya Sitthi-Amorn, and Wojciech Matusik. Design and fabrication by example. *ACM Transactions on Graphics (Proceedings SIGGRAPH 2014)*, 33(4), 2014.

[Schwartzburg and Pauly, 2013] Yuliy Schwartzburg and Mark Pauly. Fabrication-aware design with intersecting planar pieces. *Computer Graphics Forum (EUROGRAPHICS '13)*, 32(2):317–326, 2013.

[Secord *et al.*, 2011] Adrian Secord, Jingwan Lu, Adam Finkelstein, Manish Singh, and Andrew Nealen. Perceptual models of viewpoint preference. *ACM Trans. Graph.*, 2011.

[Shabana, 1990] AA Shabana. Dynamics of flexible bodies using generalized newton-euler equations. *Journal of Dynamic Systems, Measurement, and Control*, 112(3):496–503, 1990.

[Shamir, 2008] Ariel Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556, 2008.

[Shao and Badler, 1996] Min-Zhi Shao and Norman Badler. Spherical sampling by archimedes' theorem. Technical report, Univ. of Penn., 1996.

[Shneiderman, 2007] Ben Shneiderman. Creativity support tools: Accelerating discovery and innovation. *Commun. ACM*, 50(12):20–32, December 2007.

[Shoemake, 1992] Ken Shoemake. Uniform random rotations. In *Graphics Gems III*. Morgan Kaufmann, 1992.

[Sidabraite and Masteikaite, 2002] V. Sidabraite and V. Masteikaite. A preliminary study for evaluation of skirt asymmetric drape. *International Journal of Clothing Science and Technology*, 14(5):286–298, 2002.

[Skouras *et al.*, 2012a] Mélina Skouras, Bernhard Thomaszewski, Bernd Bickel, and Markus Gross. Computational design of rubber balloons. *Comput. Graphics Forum (Proc. Eurographics)*, 2012.

[Skouras *et al.*, 2012b] Mélina Skouras, Bernhard Thomaszewski, Bernd Bickel, and Markus Gross. Computational design of rubber balloons. *Comput. Graph. Forum*, 2012.

[Skouras *et al.*, 2013] Mélina Skouras, Bernhard Thomaszewski, Stelian Coros, Bernd Bickel, and Markus Gross. Computational design of actuated deformable characters. *ACM Trans. Graph.*, 2013.

[Snibbe, 1995] Scott Sona Snibbe. A direct manipulation interface for 3d computer animation. *Comput. Graph. Forum*, 1995.

[Sternberg, 1999] Robert J. Sternberg. Handbook of creativity. 1999.

[Sun and Zheng, 2015] Timothy Sun and Changxi Zheng. Computational design of twisty joints and puzzles. *ACM Trans. Graph.*, 2015.

[Talton *et al.*, 2009] Jerry O. Talton, Daniel Gibson, Lingfeng Yang, Pat Hanrahan, and Vladlen Koltun. Exploratory modeling with collaborative design spaces. *ACM Trans. Graph.*, 28(5):167:1–167:10, December 2009.

[Talton *et al.*, 2011] Jerry O. Talton, Yu Lou, Steve Lesser, Jared Duke, Radomír Měch, and Vladlen Koltun. Metropolis procedural modeling. *ACM Trans. Graph.*, 30(2):11:1–11:14, April 2011.

[Tang *et al.*, 2011] Min Tang, Dinesh Manocha, Sung-Eui Yoon, Peng Du, Jae-Pil Heo, and Ruo-Feng Tong. Volccd: Fast continuous collision culling between deforming volume meshes. *ACM Trans. Graph.*, 30(5):111:1–111:15, October 2011.

[Terry and Mynatt, 2004] Michael Terry and Elizabeth D. Mynatt. Variation in element and action: Supporting simultaneous development of alternative solutions. In *In Proceedings of CHI 2004 (Apr 24–29*, pages 711–718. ACM Press, 2004.

[Terzopoulos and Fleischer, 1988] Demetri Terzopoulos and Kurt Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *Computer Graphics (Proceedings of SIGGRAPH 88)*, volume 22, pages 269–278, August 1988.

[Teschner *et al.*, 2004] M. Teschner, S. Kimmerle, Gabriel Zachmann, B. Heidelberger, Laks Raghupathi, A. Fuhrmann, Marie-Paule Cani, Franccois Faure, N. Magnenat-Thalmann, and W. Strasser. Collision detection for deformable objects. In *Proc. Eurographics (STAR)*, 2004.

[Thomaszewski and Wacker, 2006] Bernhard Thomaszewski and Markus Wacker. Bending Models for Thin Flexible Objects. In *WSCG Short Comm.*, 2006.

[Thomaszewski *et al.*, 2014] Bernhard Thomaszewski, Stelian Coros, Damien Gauge, Vittorio Megaro, Eitan Grinspun, and Markus Gross. Computational design of linkage-based characters. *ACM Trans. Graph.*, 2014.

[Thomson, 1996] William Thomson. *Theory of vibration with applications*. CRC Press, 1996.

[Thrun and Wegbreit, 2005] S. Thrun and B. Wegbreit. Shape from symmetry. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1824–1831 Vol. 2, Oct 2005.

[Tschebyscheff, 1878] P L Tschebyscheff. Sur la coupe des vêtements, "On the cutting of garments". *Association francaise pour l'avancement des sciences*, pages 154–155, 1878.

[Twigg and James, 2007] Christopher D Twigg and Doug L James. Many-worlds browsing for control of multibody dynamics. In *ACM Transactions on Graphics (TOG)*, volume 26, page 14. ACM, 2007.

[Twigg and James, 2008] Christopher D Twigg and Doug L James. Backward steps in rigid body simulation. *ACM Transactions on Graphics (TOG)*, 27(3):25, 2008.

[Umetani *et al.*, 2011] Nobuyuki Umetani, Danny M. Kaufman, Takeo Igarashi, and Eitan Grinspun. Sensitive couture for interactive garment editing and modeling. *ACM Trans. Graph.*, 2011.

[Umetani *et al.*, 2012a] Nobuyuki Umetani, Takeo Igarashi, and Niloy J. Mitra. Guided exploration of physically valid shapes for furniture design. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2012)*, 31(4), 2012.

[Umetani *et al.*, 2012b] Nobuyuki Umetani, Takeo Igarashi, and Niloy J. Mitra. Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph.*, 2012.

[Umetani *et al.*, 2014] Nobuyuki Umetani, Yuki Koyama, Ryan Schmidt, and Takeo Igarashi. Pteromys: Interactive design and optimization of free-formed free-flight model airplanes. *ACM Trans. Graph.*, 2014.

[van West *et al.*, 1990] B P van West, R B Pipes, and M Keefe. A Simulation of the Draping of Bidirectional Fabrics over Arbitrary Surfaces. *Journal of the Textile Institute*, 81(4):448–460, 1990.

[Vanegas *et al.*, 2012] Carlos A. Vanegas, Ignacio Garcia-Dorado, Daniel G. Aliaga, Bedrich Benes, and Paul Waddell. Inverse design of urban procedural models. *ACM Trans. Graph.*, 31(6):168:1–168:11, November 2012.

[Ventsel and Krauthammer, 2001] Eduard Ventsel and Theodor Krauthammer. *Thin Plates and Shells*. CRC Press, 2001.

[Volino and Magnenat-Thalmann, 2006a] P. Volino and N. Magnenat-Thalmann. Resolving surface collisions through intersection contour minimization. *ACM Trans. Graph.*, 2006.

[Volino and Magnenat-Thalmann, 2006b] Pascal Volino and Nadia Magnenat-Thalmann. Simple linear bending stiffness in particle systems. In *SCA*, pages 101–106, 2006.

[Voss, 1882] A Voss. Über ein neues Princip der Abbildung krummer Oberflächen. *Mathematische Annalen*, 19:1–26, 1882.

[Vouga *et al.*, 2012] Etienne Vouga, Mathias Höbinger, Johannes Wallner, and Helmut Pottmann. Design of self-supporting surfaces. *ACM Trans. Graph. (SIGGRAPH '12)*, 31(4):87:1–87:11, 2012.

[Wang and Pipkin, 1986] Wen B Wang and A C Pipkin. Inextensible networks with bending stiffness. *Quarterly Journal of Mechanics & Applied Mathematics*, 39(3):343–359, 1986.

[Wang *et al.*, 1999] J Wang, R Paton, and J R Page. The draping of woven fabric preforms and prepregs for production of polymer composite components. *Composites Part A: Applied Science and Manufacturing*, 30(6):757–765, 1999.

[Wang *et al.*, 2008] W. Wang, Y. Liu, D. Yan, B. Chan, R. Ling, and F. Sun. Hexagonal meshes with planar faces. *HKU CS Tech Report TR-2008-13*, 2008.

[Wang *et al.*, 2010] Huamin Wang, Florian Hecht, Ravi Ramamoorthi, and James F O'Brien. Example-based wrinkle synthesis for clothing animation. *ACM Transactions on Graphics (TOG)*, 29(4):107, 2010.

[Wang *et al.*, 2012] Bin Wang, Francois Faure, and Dinesh K. Pai. Adaptive image-based intersection volume. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 31(4), 2012.

[Wardetzky *et al.*,] Max Wardetzky, Miklos Bergou, David Harmon, Denis Zorin, and Eitan Grinspun. Discrete Quadratic Bending Energies. To appear in CAGD, 2007.

[Weld and Leu, 1990] John D. Weld and Ming C. Leu. Geometric representation of swept volumes with application to polyhedral objects. *Int. J. Rob. Res.*, 1990.

[Whiting *et al.*, 2012] Emily Whiting, Hijung Shin, Robert Wang, John Ochsendorf, and Frédo Durand. Structural optimization of 3d masonry buildings. *ACM Trans. Graph.*, 31(6):159:1–159:11, November 2012.

[Witkin and Kass, 1988] Andrew Witkin and Michael Kass. Spacetime constraints. *Proc. SIGGRAPH*, 1988.

[Wunderlich, 1951] W Wunderlich. *Zur Differenzengeometrie der Flächen konstanter negativer Krümmung*. Springer Verlag, 1951.

[Xia and Nadler, 2011] Weijie Xia and Ben Nadler. Three-scale modeling and numerical simulations of fabric materials. *International Journal of Engineering Science*, 49(3):229–239, 2011.

[Xin *et al.*, 2011] Shiqing Xin, Chi-Fu Lai, Chi-Wing Fu, Tien-Tsin Wong, Ying He, and Daniel Cohen-Or. Making burr puzzles from 3d models. *ACM Trans. Graph. (SIGGRAPH '11)*, 30(4):97:1–97:8, July 2011.

[Yang *et al.*, 2011] Yong-Liang Yang, Yi-Jun Yang, Helmut Pottmann, and Niloy J. Mitra. Shape space exploration of constrained meshes. *ACM Transactions on Graphics*, 30(6):to appear, 2011.

[Ye and Daghyani, 1997] Lin Ye and Hamid Reza Daghyani. Characteristics of woven fibre fabric reinforced composites in forming process. *Composites Part A: Applied Science and Manufacturing*, 28(9):869–874, 1997.

[Yu *et al.*, 2011] Lap-Fai Yu, Sai-Kit Yeung, Chi-Keung Tang, Demetri Terzopoulos, Tony F. Chan, and Stanley J. Osher. Make it home: Automatic optimization of furniture arrangement. *ACM Trans. Graph.*, 30(4):86:1–86:12, July 2011.

[Zeleznik *et al.*, 2006] Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. Sketch: An interface for sketching 3d scenes. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, New York, NY, USA, 2006. ACM.

[Zhao *et al.*, 2012] Xin Zhao, Cheng-Cheng Tang, Yong-Liang Yang, and Helmut Pottmann. Intuitive design exploration of constrained meshes. In Lars Hesselgren et al., editors, *Advances in Architectural Geometry 2012*, pages 305–318. Springer, 2012.

[Zheng *et al.*, 2013] Youyi Zheng, Daniel Cohen-Or, and Niloy J. Mitra. Smart variations: Functional substructures for part compatibility. *Computer Graphics Forum (Eurographics)*, 32(2pt2):195–204, 2013.

[Zhou *et al.*, 2014] Yahan Zhou, Shinjiro Sueda, Wojciech Matusik, and Ariel Shamir. Boxelization: Folding 3d objects into boxes. *ACM Trans. Graph.*, 2014.

[Zienkiewicz and Taylor, 1989] O. C. Zienkiewicz and R. C. Taylor. *The finite element method*. McGraw Hill, 1989. 1.